Package 'BioMedR'

June 11, 2017

Type Package

Version 1.0.0

Date 2017-03-28

Title Generating Various Molecular Representations for Chemicals, Proteins, DNAs/RNAs and Their Interactions

Description The BioMedR package offers an R/Bioconductor package generating various molecular representations for chemicals, proteins, DNAs/RNAs and their interactions.

Author Min-feng Zhu <wind2zhu@163.com>, Jie Dong <biomed@csu.edu.cn>, Dong-sheng Cao <oriental-cds@163.com>

Maintainer Min-feng Zhu <wind2zhu@163.com>

License Artistic-2.0

SystemRequirements Java JDK 1.8 or higher

URL https://github.com/wind22zhu/BioMedR

BugReports https://github.com/wind22zhu/BioMedR/issues

LazyData yes

Imports RCurl, rjson, rcdk (>= 3.3.2), foreach, doParallel, Biostrings, GOSemSim, ChemmineR, fmcsR, pls, randomForest, utils, stats, graphics, methods, org.Hs.eg.db, ChemmineOB

Suggests RUnit, BiocGenerics

biocViews Software, DataImport, DataRepresentation, FeatureExtraction, Cheminformatics, BiomedicalInformatics, Proteomics, GO, GraphAndNetwork, SystemsBiology

NeedsCompilation no

R topics documented:

BioMedR-package *Toolkit for Compound-Protein Interaction in Drug Discovery*

Description

The BioMedR package offers an R/Bioconductor package emphasizing the comprehensive integration of bioinformatics and chemoinformatics into a molecular informatics platform for drug discovery.

Details

The comprehensive user guide could be opened with `vignette('BioMedR')`, which explains the functionalities included in this package in detail. The BioMedR package is developed by Computational Biology and Drug Design (CBDD) Group, Central South University (http://cbdd.csu.edu.cn/).

Package:	BioMedR
Type:	Package
Version:	Release 3
License:	Artistic-2.0

Note

Bug reports and feature requests should be sent to https://github.com/wind22zhu/BioMedR/issues.

Author(s)

Minfeng Zhu <<wind2zhu@163.com.com>> Dongsheng Cao <<oriental-cds@163.com>>

Examples

```
NULL
```

AA2DACOR *2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon*

Description

2D Autocorrelations Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AA2DACOR)
```

Details

This dataset includes the 2D autocorrelations descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AA2DACOR)
```

AA3DMoRSE *3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon*

Description

3D-MoRSE Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AA3DMoRSE)
```

Details

This dataset includes the 3D-MoRSE descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AA3DMoRSE)
```

AAACF *Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon*

Description

Atom-Centred Fragments Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAACF)
```

Details

This dataset includes the atom-centred fragments descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAACF)
```

AABLOSUM100 *BLOSUM100 Matrix for 20 Amino Acids*

Description

BLOSUM100 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM100)
```

Details

BLOSUM100 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Examples

```
data(AABLOSUM100)
```

AABLOSUM45 *BLOSUM45 Matrix for 20 Amino Acids*

Description

BLOSUM45 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM45)
```

Details

BLOSUM45 Matrix for the 20 amino acids. The matrix was extracted from the Biostrings package of Bioconductor.

Examples

```
data(AABLOSUM45)
```

AABLOSUM50 *BLOSUM50 Matrix for 20 Amino Acids*

Description

BLOSUM50 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM50)
```

Details

BLOSUM50 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM50)
```

AABLOSUM62 *BLOSUM62 Matrix for 20 Amino Acids*

Description

BLOSUM62 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM62)
```

Details

BLOSUM62 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM62)
```

AABLOSUM80 *BLOSUM80 Matrix for 20 Amino Acids*

Description

BLOSUM80 Matrix for 20 Amino Acids

Usage

```
data(AABLOSUM80)
```

Details

BLOSUM80 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AABLOSUM80)
```

AABurden *Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon*

Description

Burden Eigenvalues Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AABurden)
```

Details

This dataset includes the Burden eigenvalues descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AABurden)
```

AAConn — *Connectivity Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Connectivity Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAConn)
```

Details

This dataset includes the connectivity indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAConn)
```

AAConst — *Constitutional Descriptors for 20 Amino Acids calculated by Dragon*

Description

Constitutional Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAConst)
```

Details

This dataset includes the constitutional descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAConst)
```

AACPSA *CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio*

Description

CPSA Descriptors for 20 Amino Acids calculated by Discovery Studio

Usage

```
data(AACPSA)
```

Details

This dataset includes the CPSA descriptors of the 20 amino acids calculated by Discovery Studio (version 2.5) used for scales extraction in this package. All amino acid molecules had also been optimized with MOE 2011.10 (semiempirical AM1) before calculating these CPSA descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See `OptAA3d` for more information.

Examples

```
data(AACPSA)
```

AADescAll *All 2D Descriptors for 20 Amino Acids calculated by Dragon*

Description

All 2D Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AADescAll)
```

Details

This dataset includes all the 2D descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AADescAll)
```

AAEdgeAdj — *Edge Adjacency Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Edge Adjacency Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAEdgeAdj)
```

Details

This dataset includes the edge adjacency indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAEdgeAdj)
```

AAEigIdx — *Eigenvalue-Based Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Eigenvalue-Based Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAEigIdx)
```

Details

This dataset includes the eigenvalue-based indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAEigIdx)
```

AAFGC *Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon*

Description

Functional Group Counts Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAFGC)
```

Details

This dataset includes the functional group counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAFGC)
```

AAGeom *Geometrical Descriptors for 20 Amino Acids calculated by Dragon*

Description

Geometrical Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAGeom)
```

Details

This dataset includes the geometrical descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAGeom)
```

AAGETAWAY *GETAWAY Descriptors for 20 Amino Acids calculated by Dragon*

Description

GETAWAY Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAGETAWAY)
```

Details

This dataset includes the GETAWAY descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAGETAWAY)
```

AAindex *AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids*

Description

AAindex Data of 544 Physicochemical and Biological Properties for 20 Amino Acids

Usage

```
data(AAindex)
```

Details

The data was extracted from the AAindex1 database ver 9.1 (ftp://ftp.genome.jp/pub/db/community/aaindex/aaindex1) as of Nov. 2012 (Data Last Modified 2006-08-14).

With this data, users could investigate each property's accession number and other details. Visit http://www.genome.jp/dbget/aaindex.html for more information.

Examples

```
data(AAindex)
```

AAInfo — *Information Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Information Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAInfo)
```

Details

This dataset includes the information indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAInfo)
```

AAMetaInfo — *Meta Information for the 20 Amino Acids*

Description

Meta Information for the 20 Amino Acids

Usage

```
data(AAMetaInfo)
```

Details

This dataset includes the meta information of the 20 amino acids used for the 2D and 3D descriptor calculation in this package. Each column represents:

- `AAName` Amino Acid Name
- `Short` One-Letter Representation
- `Abbreviation` Three-Letter Representation
- `mol` SMILE Representation
- `PUBCHEM_COMPOUND_CID` PubChem CID for the Amino Acid
- `PUBCHEM_LINK` PubChem Link for the Amino Acid

Examples

```
data(AAMetaInfo)
```

AAMOE2D *2D Descriptors for 20 Amino Acids calculated by MOE 2011.10*

Description

2D Descriptors for 20 Amino Acids calculated by MOE 2011.10

Usage

```
data(AAMOE2D)
```

Details

This dataset includes the 2D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package.

Examples

```
data(AAMOE2D)
```

AAMOE3D *3D Descriptors for 20 Amino Acids calculated by MOE 2011.10*

Description

3D Descriptors for 20 Amino Acids calculated by MOE 2011.10

Usage

```
data(AAMOE3D)
```

Details

This dataset includes the 3D descriptors of the 20 amino acids calculated by MOE 2011.10 used for scales extraction in this package. All amino acid molecules had also been optimized with MOE (semiempirical AM1) before calculating these 3D descriptors. The SDF file containing the information of the optimized amino acid molecules is included in this package. See OptAA3d for more information.

Examples

```
data(AAMOE3D)
```

AAMolProp — *Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon*

Description

Molecular Properties Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAMolProp)
```

Details

This dataset includes the molecular properties descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAMolProp)
```

AAPAM120 — *PAM120 Matrix for 20 Amino Acids*

Description

PAM120 Matrix for 20 Amino Acids

Usage

```
data(AAPAM120)
```

Details

PAM120 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM120)
```

AAPAM250 *PAM250 Matrix for 20 Amino Acids*

Description

PAM250 Matrix for 20 Amino Acids

Usage

```
data(AAPAM250)
```

Details

PAM250 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM250)
```

AAPAM30 *PAM30 Matrix for 20 Amino Acids*

Description

PAM30 Matrix for 20 Amino Acids

Usage

```
data(AAPAM30)
```

Details

PAM30 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM30)
```

AAPAM40 *PAM40 Matrix for 20 Amino Acids*

Description

PAM40 Matrix for 20 Amino Acids

Usage

```
data(AAPAM40)
```

Details

PAM40 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM40)
```

AAPAM70 *PAM70 Matrix for 20 Amino Acids*

Description

PAM70 Matrix for 20 Amino Acids

Usage

```
data(AAPAM70)
```

Details

PAM70 Matrix for the 20 amino acids. The matrix was extracted from the `Biostrings` package of Bioconductor.

Examples

```
data(AAPAM70)
```

AARandic — *Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon*

Description

Randic Molecular Profiles Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AARandic)
```

Details

This dataset includes the Randic molecular profiles descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AARandic)
```

AARDF — *RDF Descriptors for 20 Amino Acids calculated by Dragon*

Description

RDF Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AARDF)
```

Details

This dataset includes the RDF descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AARDF)
```

AATopo — *Topological Descriptors for 20 Amino Acids calculated by Dragon*

Description

Topological Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AATopo)
```

Details

This dataset includes the topological descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AATopo)
```

AATopoChg — *Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon*

Description

Topological Charge Indices Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AATopoChg)
```

Details

This dataset includes the topological charge indices descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AATopoChg)
```

AAWalk — *Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon*

Description

Walk and Path Counts Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAWalk)
```

Details

This dataset includes the walk and path counts descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAWalk)
```

AAWHIM — *WHIM Descriptors for 20 Amino Acids calculated by Dragon*

Description

WHIM Descriptors for 20 Amino Acids calculated by Dragon

Usage

```
data(AAWHIM)
```

Details

This dataset includes the WHIM descriptors of the 20 amino acids calculated by Dragon (version 5.4) used for scales extraction in this package.

Examples

```
data(AAWHIM)
```

acc *Auto Cross Covariance (ACC) for Generating Scales-Based Descriptors of the Same Length*

Description

Auto Cross Covariance (ACC) for Generating Scales-Based Descriptors of the Same Length

Usage

```
acc(mat, lag)
```

Arguments

`mat`	A `p * n` matrix. Each row represents one scale (total `p` scales), each column represents one amino acid position (total `n` amino acids).
`lag`	The lag parameter. Must be less than the amino acids.

Details

This function calculates the auto covariance and auto cross covariance for generating scale-based descriptors of the same length.

Value

A length `lag * p^2` named vector, the element names are constructed by: the scales index (crossed scales index) and lag index.

Note

To know more details about auto cross covariance, see the references.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Wold, S., Jonsson, J., Sj\"orstr\"om, M., Sandberg, M., & R\"annar, S. (1993). DNA and peptide sequences and chemical processes multivariately modelled by principal component analysis and partial least-squares projections to latent structures. *Analytica chimica acta*, 277(2), 239–253.

Sj\"ostr\"om, M., R\"annar, S., & Wieslander, A. (1995). Polypeptide sequence property relationships in *Escherichia coli* based on auto cross covariances. *Chemometrics and intelligent laboratory systems*, 29(2), 295–305.

See Also

See `extrPCMScales` for generalized scales-based descriptors. For more details, see `extrPCMDescScales` and `extrPCMPropScales`.

Examples

```
p = 8    # p is the scales number
n = 200  # n is the amino acid number
lag = 7  # the lag paramter
mat = matrix(rnorm(p * n), nrow = p, ncol = n)
acc(mat, lag)
```

---

apfp *Frequent Atom Pairs*

Description

Frequent Atom Pairs

Usage

```
data(apfp)
```

Details

Object stores 4096 most frequent atom pairs generated from DrugBank compounds.

Examples

```
data(apfp)
```

---

atomprop *Standard atomic weights*

Description

Standard atomic weights

Usage

```
data(atomprop)
```

Details

Data frame with atom names, symbols, standard atomic weights, group number and period number.

Examples

```
data(atomprop)
```

Autocorrelation *Calculates the Moreau-Broto Autocorrelation Descriptors using Partial Charges*

Description

Calculates the Moreau-Broto Autocorrelation Descriptors using Partial Charges

Calculates the Moreau-Broto Autocorrelation Descriptors using Atomic Weight

Calculates the Moreau-Broto Autocorrelation Descriptors using Polarizability

Usage

```
extrDrugAutocorrelationcharge(molecules, silent = TRUE)

extrDrugAutocorrelationMass(molecules, silent = TRUE)

extrDrugAutocorrelationPolarizability(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the ATS autocorrelation descriptor, where the weight equal to the charges.

Calculates the ATS autocorrelation descriptor, where the weight equal to the scaled atomic mass.

Calculates the ATS autocorrelation descriptor using polarizability.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 5 columns named ATSc1, ATSc2, ATSc3, ATSc4, ATSc5.

extrDrugAutocorrelationMass: This function returns 5 columns named ATSm1, ATSm2, ATSm3, ATSm4, ATSm5.

extrDrugAutocorrelationPolarizability: This function returns 5 columns named ATSp1, ATSp2, ATSp3, ATSp4, ATSp5.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Moreau, Gilles, and Pierre Broto. The autocorrelation of a topological structure: a new molecular descriptor. Nouv. J. Chim 4 (1980): 359-360.

Examples

```
# Calculates the Moreau-Broto Autocorrelation Descriptors using Partial Charges
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugAutocorrelationcharge(mol)
head(dat)
# Calculates the Moreau-Broto Autocorrelation Descriptors using Atomic Weight
dat = extrDrugAutocorrelationMass(mol)
head(dat)
# Calculates the Moreau-Broto Autocorrelation Descriptors using Polarizability
dat = extrDrugAutocorrelationPolarizability(mol)
head(dat)
```

---

bcl *2D descriptors of bcl2*

Description

2D descriptors of bcl2

Usage

```
data(bcl)
```

Details

object stores 380 molecules and 160 2D descriptors

Examples

```
data(bcl)
```

BMgetDNAGenBank *Get DNA/RNA Sequences from Genbank by GI ID*

Description

Get DNA/RNA Sequences from Genbank by GI ID

Usage

```
BMgetDNAGenBank(id)
```

Arguments

id	A character vector, as the GI ID(s).

Details

This function get DNA/RNA sequences from Genbank by GI ID(s).

Value

A list, each component contains one of the DNA/RNA sequences.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See readFASTA for reading FASTA format files.

Examples

```
# Network latency may slow down this example
# Only test this when your connection is fast enough

ids = c(2, 11)
BMgetDNAGenBank(ids)
```

---

calcDrugFPSim *Calculate Drug Molecule Similarity Derived by Molecular Fingerprints*

Description

Calculate Drug Molecule Similarity Derived by Molecular Fingerprints

Usage

```
calcDrugFPSim(fp1, fp2, fptype = c("compact", "complete"),
  metric = c("tanimoto", "euclidean", "cosine", "dice", "hamming"))
```

Arguments

fp1	The first molecule's fingerprints, could be extracted by extractDrugMACCS(), extractDrugMACCSComplete() etc.
fp2	The second molecule's fingerprints.
fptype	The fingerprint type, must be one of "compact" or "complete".
metric	The similarity metric, one of "tanimoto", "euclidean", "cosine", "dice" and "hamming".

Details

This function calculate drug molecule fingerprints similarity. Define a as the features of object A, b is the features of object B, c is the number of common features to A and B:

- Tanimoto: aka Jaccard - $c/a+b+c$
- Euclidean: $\sqrt{(a+b)}$
- Dice: aka Sorensen, Czekanowski, Hodgkin-Richards - $c/0.5[(a+c)+(b+c)]$
- Cosine: aka Ochiai, Carbo - $c/\sqrt{((a+c)(b+c))}$
- Hamming: aka Manhattan, taxi-cab, city-block distance - $(a+b)$

Value

The numeric similarity value.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Gasteiger, Johann, and Thomas Engel, eds. Chemoinformatics. Wiley.com, 2006.

Examples

```
mols = readMolFromSDF(system.file('compseq/tyrphostin.sdf', package = 'BioMedR'))

fp1 = extrDrugEstate(mols[[1]])
fp2 = extrDrugEstate(mols[[2]])
calcDrugFPSim(fp1, fp2, fptype = 'compact', metric = 'tanimoto')
```

---

calcDrugMCSSim *Calculate Drug Molecule Similarity Derived by Maximum Common Substructure Search*

Description

Calculate Drug Molecule Similarity Derived by Maximum Common Substructure Search

Usage

```
calcDrugMCSSim(mol1, mol2, type = c("smile", "sdf"), plot = FALSE, al = 0,
  au = 0, bl = 0, bu = 0, matching.mode = "static", ...)
```

Arguments

mol1	The first molecule. R character string object containing the molecule. See examples.
mol2	The second molecule. R character string object containing the molecule. See examples.
type	The input molecule format, 'smile' or 'sdf'.
plot	Logical. Should we plot the two molecules and their maximum common substructure?
al	Lower bound for the number of atom mismatches. Default is 0.
au	Upper bound for the number of atom mismatches. Default is 0.
bl	Lower bound for the number of bond mismatches. Default is 0.
bu	Upper bound for the number of bond mismatches. Default is 0.
matching.mode	Three modes for bond matching are supported: 'static', 'aromatic', and 'ring'.
...	Other graphical parameters

Details

This function calculate drug molecule similarity derived by maximum common substructure search. The maximum common substructure search algorithm is provided by the `fmcsR` package.

Value

A list containing the detail MCS information and similarity values. The numeric similarity value includes Tanimoto coefficient and overlap coefficient.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Wang, Y., Backman, T. W., Horan, K., & Girke, T. (2013). fmcsR: mismatch tolerant maximum common substructure searching in R. Bioinformatics, 29(21), 2792–2794.

Examples

```
mol1 = 'CC(C)CCCCCC(=O)NCC1=CC(=C(C=C1)O)OC'
mol2 = 'O=C(NCc1cc(OC)c(O)cc1)CCCC/C=C/C(C)C'
sim1 = calcDrugMCSSim(mol1, mol2, type = 'smile')
print(sim1[[2]])  # Tanimoto Coefficient
```

`calcParProtGOSim` *Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein Sequence Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
calcParProtGOSim(golist, type = c("go", "gene"), ont = "MF",
  organism = "human", measure = "Resnik", combine = "BMA")
```

Arguments

`golist`	A character vector, each component contains a character vector of GO terms or one Entrez Gene ID.
`type`	Input type of `golist`, `'go'` for GO Terms, `'gene'` for gene ID.
`ont`	Default is `'MF'`, could be one of `'MF'`, `'BP'`, or `'CC'` subontologies.
`organism`	Default is `'human'`, could be one of `'anopheles'`, `'arabidopsis'`, `'bovine'`, `'canine'`, `'chicken'`, `'chimp'`, `'coelicolor'`, `'ecolik12'`, `'ecsakai'`, `'fly'`, `'human'`, `'malaria'`, `'mouse'`, `'pig'`, `'rat'`, `'rhesus'`, `'worm'`, `'xenopus'`, `'yeast'` or `'zebrafish'`.
`measure`	Default is `'Resnik'`, could be one of `'Resnik'`, `'Lin'`, `'Rel'`, `'Jiang'` or `'Wang'`.
`combine`	Default is `'BMA'`, could be one of `'max'`, `'average'`, `'rcmax'` or `'BMA'` for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates protein sequence similarity based on Gene Ontology (GO) similarity.

Value

A n x n similarity matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See calcTwoProtGOSim for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs. See calcParProtSeqSim for paralleled protein similarity calculation based on sequence alignment.

Examples

```
# by GO Terms
go1 = c('GO:0005215', 'GO:0005488', 'GO:0005515', 'GO:0005625', 'GO:0005802', 'GO:0005905')  # AP4B1
go2 = c('GO:0005515', 'GO:0005634', 'GO:0005681', 'GO:0008380', 'GO:0031202')  # BCAS2
go3 = c('GO:0003735', 'GO:0005622', 'GO:0005840', 'GO:0006412')  # PDE4DIP
glist = list(go1, go2, go3)
gsimmat1 = calcParProtGOSim(glist, type = 'go', ont = 'CC')
print(gsimmat1)

# by Entrez gene id
genelist = list(c('150', '151', '152', '1814', '1815', '1816'))
gsimmat2 = calcParProtGOSim(genelist, type = 'gene')
print(gsimmat2)
```

---

calcParProtSeqSim *Parallellized Protein Sequence Similarity Calculation based on Sequence Alignment*

Description

Parallellized Protein Sequence Similarity Calculation based on Sequence Alignment

Usage

```
calcParProtSeqSim(protlist, cores = 2, type = "local",
  submat = "BLOSUM62")
```

Arguments

protlist	A length n list containing n protein sequences, each component of the list is a character string, storing one protein sequence. Unknown sequences should be represented as ''.
cores	Integer. The number of CPU cores to use for parallel execution, default is 2. Users could use the detectCores() function in the parallel package to see how many cores they could use.
type	Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.
submat	Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implemented the parallellized version for calculating protein sequence similarity based on sequence alignment.

Value

A n x n similarity matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See calcTwoProtSeqSim for protein sequence alignment for two protein sequences. See calcParProtGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity.

Examples

```
s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
s2 = readFASTA(system.file('protseq/P08218.fasta', package = 'BioMedR'))[[1]]
s3 = readFASTA(system.file('protseq/P10323.fasta', package = 'BioMedR'))[[1]]
s4 = readFASTA(system.file('protseq/P20160.fasta', package = 'BioMedR'))[[1]]
s5 = readFASTA(system.file('protseq/Q9NZP8.fasta', package = 'BioMedR'))[[1]]
plist = list(s1, s2, s3, s4, s5)
psimmat = calcParProtSeqSim(plist, cores = 2, type = 'local', submat = 'BLOSUM62')
```

---

calcTwoProtGOSim *Protein Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
calcTwoProtGOSim(id1, id2, type = c("go", "gene"), ont = "MF",
  organism = "human", measure = "Resnik", combine = "BMA")
```

Arguments

id1	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
id2	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
type	Input type of id1 and id2, 'go' for GO Terms, 'gene' for gene ID.
ont	Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.
organism	Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecsakai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.
measure	Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.
combine	Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates the Gene Ontology (GO) similarity between two groups of GO terms or two Entrez gene IDs.

Value

A n x n matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See calcParProtGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity. See calcParProtSeqSim for paralleled protein similarity calculation based on sequence alignment.

Examples

```
# by GO terms
go1 = c("GO:0004022", "GO:0004024", "GO:0004023")
go2 = c("GO:0009055", "GO:0020037")
gsim1 = calcTwoProtGOSim(go1, go2, type = 'go', ont = 'MF', measure = 'Wang')
print(gsim1)

# by Entrez gene id
gene1 = '241'
```

```
gene2 = '251'
gsim2 = calcTwoProtGOSim(gene1, gene2, type = 'gene', ont = 'CC', measure = 'Lin')
print(gsim2)
```

`calcTwoProtSeqSim` *Protein Sequence Alignment for Two Protein Sequences*

Description

Protein Sequence Alignment for Two Protein Sequences

Usage

```
calcTwoProtSeqSim(seq1, seq2, type = "local", submat = "BLOSUM62")
```

Arguments

`seq1`	A character string, containing one protein sequence.
`seq2`	A character string, containing another protein sequence.
`type`	Type of alignment, default is `'local'`, could be `'global'` or `'local'`, where `'global'` represents Needleman-Wunsch global alignment; `'local'` represents Smith-Waterman local alignment.
`submat`	Substitution matrix, default is `'BLOSUM62'`, could be one of `'BLOSUM45'`, `'BLOSUM50'`, `'BLOSUM62'`, `'BLOSUM80'`, `'BLOSUM100'`, `'PAM30'`, `'PAM40'`, `'PAM70'`, `'PAM120'`, `'PAM250'`.

Details

This function implements the sequence alignment between two protein sequences.

Value

An Biostrings object containing the scores and other alignment information.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `calcParProtSeqSim` for paralleled pairwise protein similarity calculation based on sequence alignment. See `calcTwoProtGOSim` for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs.

Examples

```
s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
s2 = readFASTA(system.file('protseq/P10323.fasta', package = 'BioMedR'))[[1]]
seqalign = calcTwoProtSeqSim(s1, s2)
summary(seqalign)
```

checkDNA *Check if the DNA sequence are in the 4 default types*

Description

Check if the DNA sequence are in the 4 default types

Usage

```
checkDNA(x)
```

Arguments

x	A character vector, as the input DNA sequence.

Details

This function checks if the DNA sequence types are in the 4.

Value

Logical. TRUE if all of the DNA types of the sequence are within the 4 default types.

The result character vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
checkDNA(x) # TRUE
checkDNA(paste(x, 'Z', sep = '')) # FALSE
```

checkProt *Check if the protein sequence's amino acid types are the 20 default types*

Description

Check if the protein sequence's amino acid types are the 20 default types

Usage

```
checkProt(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function checks if the protein sequence's amino acid types are the 20 default types.

Value

Logical. TRUE if all of the amino acid types of the sequence are within the 20 default types.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
checkProt(x)  # TRUE
checkProt(paste(x, 'Z', sep = ''))  # FALSE
```

---

clusterCMP *cluster compounds using a descriptor database*

Description

'clusterCMP' uses structural compound descriptors and clusters the compounds based on their pairwise distances. clusterCMP uses single linkage to measure distance between clusters when it merges clusters. It accepts both a single cutoff and a cutoff vector. By using a cutoff vector, it can generate results similar to hierarchical clustering after tree cutting.

Usage

```
clusterCMP(db, cutoff, is.similarity = TRUE, save.distances = FALSE,
  use.distances = NULL, quiet = FALSE, ...)
```

Arguments

db	The desciptor database
cutoff	The clustering cutoff. Can be a single value or a vector. The cutoff gives the maximum distance between two compounds in order to group them in the same cluster.
is.similarity	Set when the cutoff supplied is a similarity cutoff. This cutoff is the minimum similarity value between two compounds such that they will be grouped in the same cluster.
save.distances	whether to save distance for future clustering. See details below.
use.distances	Supply pre-computed distance matrix.
quiet	Whether to suppress the progress information.
...	Further arguments to be passed to similarity.

Details

clusterCMP will compute distances on the fly if use.distances is not set. Furthermore, if save.distances is not set, the distance values computed will never be stored and any distance between two compounds is guaranteed not to be computed twice. Using this method, clusterCMP can deal with large databases when a distance matrix in memory is not feasible. The speed of the clustering function should be slowed when using a transient distance calculation. When save.distances is set, clusterCMP will be forced to compute the distance matrix and save it in memory before the clustering. This is useful when additional clusterings are required in the future without re-computed the distance matrix. Set save.distances to TRUE if you only want to force the clustering to use this 2-step approach; otherwise, set it to the filename under which you want the distance matrix to be saved. After you save it, when you need to reuse the distance matrix, you can 'load' it, and supply it to clusterCMP via the use.distances argument. clusterCMP supports a vector of several cutoffs. When you have multiple cutoffs, clusterCMP still guarantees that pairwise distances will never be recomputed, and no copy of distances is kept in memory. It is guaranteed to be as fast as calling clusterCMP with a single cutoff that results in the longest processing time, plus some small overhead linear in processing time.

Value

Returns a data.frame. Besides a variable giving compound ID, each of the other variables in the data frame will either give the cluster IDs of compounds under some clustering cutoff, or the size of clusters that the compounds belong to. When N cutoffs are given, in total 2*N+1 variables will be generated, with N of them giving the cluster ID of each compound under each of the N cutoffs, and the other N of them giving the cluster size under each of the N cutoffs. The rows are sorted by cluster sizes.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

...

See Also

See clusterStat for generate statistics on sizes of clusters.

Examples

```
data(sdfbcl)
apbcl <- convSDFtoAP(sdfbcl)
fpbcl <- convAPtoFP(apbcl)
clusters <- clusterCMP(db = apbcl, cutoff = c(0.5, 0.85))
clusters2 <- clusterCMP(fpbcl, cutoff = c(0.5, 0.7), method = "Tversky")
clusters <- clusterCMP(apbcl, cutoff = 0.65, save.distances = "distmat.rda")
load("distmat.rda")
clusters <- clusterCMP(apbcl, cutoff = 0.60, use.distances = distmat)
```

clusterJP *Jarvis-Patrick Clustering*

Description

Jarvis-Patrick Clustering

Usage

```
clusterJP(nnm, k, mode = "a1a2b", linkage = "single")
```

Arguments

nnm	A nearest neighbor table, as produced by nearestNeighbors.
k	Minimum number of nearest neighbors two rows (items) in the nearest neighbor table need to have in common to join them into the same cluster.
mode	If mode = "a1a2b" (default), the clustering is run with both requirements (a) and (b); if mode = "a1b" then (a) is relaxed to a unidirectional requirement; and if mode = "b" then only requirement (b) is used. The size of the clusters generated by the different methods increases in this order: "a1a2b" < "a1b" < "b". The run time of method "a1a2b" follows a close to linear relationship, while it is nearly quadratic for the much more exhaustive method "b". Only methods "a1a2b" and "a1b" are suitable for clustering very large data sets (e.g. >50,000 items) in a reasonable amount of time.
linkage	Can be one of "single", "average", or "complete", for single linkage, average linkage and complete linkage merge requirements, respectively. In the context of Jarvis-Patrick, average linkage means that at least half of the pairs between the clusters under consideration must pass the merge requirement. Similarly, for complete linkage, all pairs must pass the merge requirement. Single linkage is the normal case for Jarvis-Patrick and just means that at least one pair must meet the requirement.

Details

Function to perform Jarvis-Patrick clustering. The algorithm requires a nearest neighbor table, which consists of neighbors for each item in the dataset. This information is then used to join items into clusters with the following requirements: (a) they are contained in each other's neighbor list (b) they share at least k nearest neighbors The nearest neighbor table can be computed with NNeighbors. For standard Jarvis-Patrick clustering, this function takes the number of neighbors to keep for each item.

Value

Depending on the setting under the type argument, the function returns the clustering result in a named vector or a nearest neighbor table as matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Jarvis RA, Patrick EA (1973) Clustering Using a Similarity Measure Based on Shared Near Neighbors. IEEE Transactions on Computers, C22, 1025-1034. URLs: http://davide.eynard.it/teaching/2012_PAMI/JP.pdf, http://www.btluke.com/jpclust.html, http://www.daylight.com/dayhtml/doc/cluster/index.pdf

See Also

see NNeighbors for nearest neighbors

Examples

```
data(sdfbcl)
apbcl <- convSDFtoAP(sdfbcl)
fpbcl <- convAPtoFP(apbcl)
clusterJP(NNeighbors(apbcl, numNbrs = 6), k = 5, mode = "a1a2b")
clusterJP(NNeighbors(fpbcl, numNbrs = 6), k = 5, mode = "a1b")
clusterJP(NNeighbors(apbcl, cutoff = 0.6), k = 2, mode = 'b')
```

---

clusterMDS *visualize clustering result using multi-dimensional scaling*

Description

'clusterMDS' takes clustering result returned by 'clusterCMP' and generate multi-dimensional scaling plot for visualization purpose.

Usage

```
clusterMDS(db, cls, size.cutoff, distmat = NULL, color.vector = NULL,
  cluster.result = 1, dimensions = 2, quiet = FALSE,
  highlight.compounds = NULL, highlight.color = NULL)
```

Arguments

db	The desciptor database
cls	The clustering result returned by 'clusterCMP'.
size.cutoff	The cutoff size for clusters considered in this visualization. Clusters of size smaller than the cutoff will not be considered.
distmat	A distance matrix that corresponds to the 'db'. If not provided, it will be computed on-the-fly in an efficient manner.
color.vector	Colors to be used in the plot. If the number of colors in the vector is not enough for the plot, colors will be reused. If not provided, color will be generated and randomly sampled from 'rainbow'.
cluster.result	Used to select the clustering result if multiple clustering results are present in 'cls'.
dimensions	Dimensionality to be used in visualization. See details.
quiet	Whether to supress the progress bar.
highlight.compounds	A vector of compound IDs, corresponding to compounds to be highlighted in the plot. A highlighted compound is represented as a filled circle.

highlight.color
: Color used for highlighted compounds. If not set, a highlighted compounds will have the same color as that used for other compounds in the same cluster.

Details

'clusterMDS' internally calls the 'cmdscale' function to generate a set of points in 2-D for the compounds in selected clusters.Note that for compounds in clusters smaller than the cutoff size, they will not be considered in this calculation - their entries in 'distmat' will be discarded if 'distmat' is provided, and distances involving them will not be computed if 'distmat' is not provided. To determine the value for 'size.cutoff', you can use 'cluster.sizestat' to see the size distribution of clusters. Because 'clusterCMP' function allows you to perform multiple clustering processes simultaneously with different cutoff values, the 'cls' parameter may point to a data frame containing multiple clustering results. The user can use 'cluster.result' to specify which result to use. By default, this is set to 1, and the first clustering result will be used in visualization. Whatever the value is, in interactive mode (described below), all clustering result will be displayed when a compound is selected in the interactive plot. If the colors provided in 'color.vector' are not enough to distinguish clusters by colors, the function will silently reuse the colors, resulting multiple clusters colored in the same color. By default, 'dimensions' is set to 2, and the built-in 'plot' function will be used for plotting. If you need to do 3-Dimensional plotting, set 'dimensions' to 3, and pass the returned value to 3D plot utilities, such as 'scatterplot3d' or 'rggobi'. This package does not perform 3D plot on its own.

Value

This function returns a data frame of MDS coordinates and clustering result. This value can be passed to 3D plot utilities such as 'scatterplot3d' and 'rggobi'.

The last column of the output gives whether the compounds have been clicked in the interactive mode.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

...

See Also

See clusterCMP for cluster compounds using a descriptor database.

Examples

```
data(sdfbcl)
apbcl = convSDFtoAP(sdfbcl)
clusters <- clusterCMP(apbcl, cutoff = c(0.5, 0.4))
clusterMDS(apbcl, clusters, size.cutoff = 2, quiet = TRUE)
```

`clusterStat` *generate statistics on sizes of clusters*

Description

'cluster.sizestat' is used to do simple statistics on sizes of clusters generated by 'clusterCMP'. It will return a dataframe which maps a cluster size to the number of clusters with that size. It is often used along with 'cluster.visualize'.

Usage

```
clusterStat(cls, cluster.result = 1)
```

Arguments

`cls`	The clustering result returned by 'clusterCMP'.
`cluster.result`	If multiple cutoff values are used in clustering process, this argument tells which cutoff value is to be considered here.

Details

'cluster.sizestat' depends on the format that is returned by 'clusterCMP' - it will treat the first column as the indecies, and the second column as the cluster sizes of effective clustering. Because of this, when multiple cutoffs are used when 'clusterCMP' is called, 'cluster.sizestat' will only consider the clustering result of the first cutoff. If you want to work on an alternative cutoff, you have to manually reorder/remove columns.

Value

Returns A data frame of two columns.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

...

Examples

```
data(sdfbcl)
apbcl <- convSDFtoAP(sdfbcl)

cluster <- clusterCMP(db = apbcl, cutoff = c(0.65, 0.5))
clusterStat(cluster[, c(1, 2, 3)])
clusterStat(cluster[, c(1, 4, 5)])
```

`connectivity` *Calculate the Kier and Hall Chi Chain Indices of Orders 3, 4, 5, 6 and 7*

Description

Calculate the Kier and Hall Chi Chain Indices of Orders 3, 4, 5, 6 and 7

Evaluates the Kier and Hall Chi cluster indices of orders 3, 4, 5 and 6

the Kier and Hall Chi Path Cluster Indices of Orders 4, 5 and 6

Calculate the Kier and Hall Chi Path Indices of Orders 0 to 7

Usage

```
extrDrugChiChain(molecules, silent = TRUE)

extrDrugChiCluster(molecules, silent = TRUE)

extrDrugChiPathCluster(molecules, silent = TRUE)

extrDrugChiPath(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Evaluates chi chain descriptors. The code currently evluates the simple and valence chi chain descriptors of orders 3, 4, 5, 6 and 7. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Evaluates chi cluster descriptors. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Evaluates chi path cluster descriptors. The code currently evluates the simple and valence chi chain descriptors of orders 4, 5 and 6. It utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Evaluates chi path descriptors. This function utilizes the graph isomorphism code of the CDK to find fragments matching SMILES strings representing the fragments corresponding to each type of chain.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 10 columns, in the following order:

- `SCH.3` - Simple chain, order 3
- `SCH.4` - Simple chain, order 4
- `SCH.5` - Simple chain, order 5

- SCH.6 - Simple chain, order 6
- SCH.7 - Simple chain, order 7
- VCH.3 - Valence chain, order 3
- VCH.4 - Valence chain, order 4
- VCH.5 - Valence chain, order 5
- VCH.6 - Valence chain, order 6
- VCH.7 - Valence chain, order 7

extrDrugChiCluster: This function returns 8 columns, the order and names of the columns returned is:

- SC.3 - Simple cluster, order 3
- SC.4 - Simple cluster, order 4
- SC.5 - Simple cluster, order 5
- SC.6 - Simple cluster, order 6
- VC.3 - Valence cluster, order 3
- VC.4 - Valence cluster, order 4
- VC.5 - Valence cluster, order 5
- VC.6 - Valence cluster, order 6

extrDrugChiPathCluster: This function returns 6 columns named SPC.4, SPC.5, SPC.6, VPC.4, VPC.5, VPC.6:

- SPC.4 - Simple path cluster, order 4
- SPC.5 - Simple path cluster, order 5
- SPC.6 - Simple path cluster, order 6
- VPC.4 - Valence path cluster, order 4
- VPC.5 - Valence path cluster, order 5
- VPC.6 - Valence path cluster, order 6

extrDrugChiPath: This function returns 16 columns, The order and names of the columns returned is:

- SP.0, SP.1, ..., SP.7 - Simple path, orders 0 to 7
- VP.0, VP.1, ..., VP.7 - Valence path, orders 0 to 7

Note

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

extrDrugChiPathCluster : These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z

use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

extrDrugChiPath: These descriptors are calculated using graph isomorphism to identify the various fragments. As a result calculations may be slow. In addition, recent versions of Molconn-Z use simplified fragment definitions (i.e., rings without branches etc.) whereas these descriptors use the older more complex fragment definitions.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
# Calculate the Kier and Hall Chi Chain Indices of Orders 3, 4, 5, 6 and 7
dat = extrDrugChiChain(mol)
head(dat)
# Evaluates the Kier and Hall Chi cluster indices of orders 3, 4, 5 and 6
dat = extrDrugChiCluster(mol)
head(dat)
# Calculate the Kier and Hall Chi Path Cluster Indices of Orders 4, 5 and 6
dat = extrDrugChiPathCluster(mol)
head(dat)
# Calculate the Kier and Hall Chi Path Indices of Orders 0 to 7
dat = extrDrugChiPath(mol)
head(dat)
```

Constitutional *Calculates the Number of Amino Acids Descriptor*

Description

Calculates the Number of Amino Acids Descriptor

Calculates the Number of Aromatic Atoms Descriptor

Calculates the Number of Aromatic Bonds Descriptor

Calculates the Number of Atom Descriptor

Calculates the Descriptor Based on the Number of Bonds of a Certain Bond Order

Descriptor that Calculates the Number of Atoms in the Largest Chain

Descriptor that Calculates the Number of Atoms in the Largest Pi Chain

Descriptor that Calculates the Number of Atoms in the Longest Aliphatic Chain

Descriptor that Calculates the Number of Nonrotatable Bonds on A Molecule

Usage

```
extrDrugAminoAcidCount(molecules, silent = TRUE)

extrDrugAromaticAtomsCount(molecules, silent = TRUE)

extrDrugAromaticBondsCount(molecules, silent = TRUE)
```

```
extrDrugAtomCount(molecules, silent = TRUE)

extrDrugBondCount(molecules, silent = TRUE)

extrDrugLargestChain(molecules, silent = TRUE)

extrDrugLargestPiSystem(molecules, silent = TRUE)

extrDrugLongestAliphaticChain(molecules, silent = TRUE)

extrDrugRotatableBondsCount(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the number of each amino acids (total 20 types) found in the molecues.

Calculates the number of aromatic atoms of a molecule.

Calculates the number of aromatic bonds of a molecule.

Calculates the number of atoms of a certain element type in a molecule. By default it returns the count of all atoms.

Calculates the descriptor based on the number of bonds of a certain bond order.

This descriptor calculates the number of atoms in the largest chain. Note that a chain exists if there are two or more atoms. Thus single atom molecules will return 0.

This descriptor calculates the number of atoms in the largest pi chain.

This descriptor calculates the number of atoms in the longest aliphatic chain.

The number of rotatable bonds is given by the SMARTS specified by Daylight on SMARTS tutorial (http://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html#EXMPL)

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 20 columns named nA, nR, nN, nD, nC, nF, nQ, nE, nG, nH, nI, nP, nL nK, nM, nS, nT, nY nV, nW.

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named naAromAtom.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
# Calculates the Number of Amino Acids Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugAminoAcidCount(mol)
head(dat)
# Calculates the Number of Aromatic Atoms Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugAromaticAtomsCount(mol)
head(dat)
# Calculates the Number of Aromatic Bonds Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugAromaticBondsCount(mol)
head(dat)
# Calculates the Number of Atom Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugAtomCount(mol)
head(dat)
# Calculates the Descriptor Based on the Number of Bonds of a
# Certain Bond Order
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugBondCount(mol)
head(dat)
# Descriptor that Calculates the Number of Atoms in the Largest Chain
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugLargestChain(mol)
head(dat)
# Descriptor that Calculates the Number of Atoms in the Largest Pi Chain
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugLargestPiSystem(mol)
head(dat)
# Descriptor that Calculates the Number of Atoms in the Longest Aliphatic Chain
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugLongestAliphaticChain(mol)
head(dat)
# Descriptor that Calculates the Number of Nonrotatable Bonds on A Molecule
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugRotatableBondsCount(mol)
head(dat)
```

---

convAPtoFP *Fingerprints from descriptor vectors*

Description

Fingerprints from descriptor vectors

Usage

```
convAPtoFP(x, descnames = 1024)
```

Arguments

x	Object of classe APset or list of vectors
descnames	Descriptor set to consider for fingerprint encoding. If a single value from 1-4096 is provided then the function uses the corresponding number of the most frequent atom pairs stored in the apfp data set provided by the package. Alternatively, one can provide here any custom atom pair selection in form of a character vector.

Details

Generates fingerprints from descriptor vectors such as atom pairs stored in APset or list containers. The obtained fingerprints can be used for structure similarity comparisons, searching and clustering. Due to their compact size, computations on fingerprints are often more time and memory efficient than on their much more complex atom pair counterparts.

Value

A FPset

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See convSDFtoAP for Atom pair library.

Examples

```
data(sdfbcl)
apbcl = convSDFtoAP(sdfbcl)
fpbcl = convAPtoFP(apbcl)
```

---

convSDFtoAP *Atom pair library*

Description

Atom pair library

Usage

```
convSDFtoAP(sdfset, type = "AP", uniquePairs = TRUE)
```

Arguments

sdfset	Objects of classes SDFset or SDF
type	if type="AP", the function returns APset/AP objects; if type="character", it returns the result as a character vector of length one. The latter is useful for storing AP data in tabular files.
uniquePairs	When the same atom pair occurs more than once in a single compound, should the names be unique or not? Setting this to true will take slightly longer to compute.

Details

Creates from a SDFset a searchable atom pair library that is stored in a container of class APset.

Value

APset	if input is SDFset
AP	if input is SDF

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>,

Examples

```
data(sdfbcl)

apset <- convSDFtoAP(sdfbcl)
```

---

extrDNADAC *The Dinucleotide-based Auto Covariance Descriptor*

Description

The Dinucleotide-based Auto Covariance Descriptor

Usage

```
extrDNADAC(x, index = c("Twist", "Tilt"), nlag = 2, normaliztion = FALSE,
  customprops = NULL, allprop = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 38 different physicochemical indices (Table 1), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.

normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
allprop	all the 38 physicochemical indices will be employed to generate the feature vector. Its default value is False.

Details

This function calculates the dinucleotide-based auto covariance descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Dong Q, Zhou S, Guan J. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 2009, 25(20): 2655-2662.

See Also

See extrDNADCC and extrDNADACC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNADAC(x)
```

---

extrDNADACC *The Dinucleotide-based Auto-cross Covariance Descriptor*

Description

The Dinucleotide-based Auto-cross Covariance Descriptor

Usage

```
extrDNADACC(x, index = c("Twist", "Tilt"), nlag = 2, normaliztion = FALSE,
  customprops = NULL, allprop = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 38 different physicochemical indices (Table 1), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.
normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
allprop	all the 38 physicochemical indices will be employed to generate the feature vector. Its default value is False.

Details

This function calculates the dinucleotide-based auto-cross covariance descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Dong Q, Zhou S, Guan J. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 2009, 25(20): 2655-2662.

See Also

See extrDNADAC and extrDNADCC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNADACC(x)
```

extrDNADCC *The Dinucleotide-based Cross Covariance Descriptor*

Description

The Dinucleotide-based Cross Covariance Descriptor

Usage

```
extrDNADCC(x, index = c("Twist", "Tilt"), nlag = 2, normaliztion = FALSE,
  customprops = NULL, allprop = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 38 different physicochemical indices (Table 1), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.
normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
allprop	all the 38 physicochemical indices will be employed to generate the feature vector. Its default value is False.

Details

This function calculates the dinucleotide-based cross covariance descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Dong Q, Zhou S, Guan J. A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. *Bioinformatics*, 2009, 25(20): 2655-2662.

See Also

See extrDNADAC and extrDNADACC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNADCC(x)
```

---

extrDNAIncDiv *The Increment Of Diversity Descriptors*

Description

The Increment Of Diversity Descriptors

Usage

```
extrDNAIncDiv(k = 6, x, pos, neg, upto = TRUE)
```

Arguments

k	the k value of kmer, it should be an integer larger than 0,the default value is 6.
x	the input data, which should be a list or file type.
pos	the positive source data, which should be a or type.
neg	the negative source data, which should be or type.
upto	generate all the kmers: 1mer, 2mer, ..., kmer. The output feature vector is the combination of all these kmers. The default value of this parameter is True

Details

This function calculates the The Basic Kmer Descriptor

Value

if upto is True, A length k * 2 named vector, k is the k value of kmer; if upto is False, A length 2 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Chen W, Luo L, Zhang L. The organization of nucleosomes around splice sites. *Nucleic acids research*, 2010, 38(9): 2788-2798. Liu G, Liu J, Cui X, et al. Sequence-dependent prediction of recombination hotspots in Saccharomyces cerevisiae. *Journal of theoretical biology*, 2012, 293: 49-54.

See Also

See extrDNAkmer

Examples

```
pos = readFASTA(system.file('dnaseq/pos.fasta', package = 'BioMedR'))
neg = readFASTA(system.file('dnaseq/neg.fasta', package = 'BioMedR'))
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNAIncDiv(k = 6, x, pos, neg)
```

`extrDNAkmer` *The Basic Kmer Descriptor*

Description

The Basic Kmer Descriptor

Usage

```
extrDNAkmer(x, k = 2, upto = FALSE, normalize = FALSE, reverse = FALSE)
```

Arguments

`x`	the input data, which should be a list or file type.
`k`	the k value of kmer, it should be an integer larger than 0.
`upto`	generate all the kmers: 1mer, 2mer, ..., kmer. The output feature vector is the combination of all these kmers. The default value of this parameter is False.
`normalize`	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
`reverse`	make reverse complements into a single feature, The default value of this parameter is False. if reverse is True, this method returns the reverse compliment kmer feature vector.

Details

This function calculates the basic kmer descriptor

Value

A vector

Note

if the parameters normalize and upto are both True, and then the feature vector is the combination of all these normalized kmers, e.g. the combination of normalized 1-kmer and normalized 2-kmer when k=2, normalize=True, upto=True.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Noble W S, Kuehn S, Thurman R, et al. Predicting the in vivo signature of human gene regulatory sequences. *Bioinformatics*, 2005, 21 Suppl 1, i338-343. Lee D, Karchin R, Beer M A. Discriminative prediction of mammalian enhancers from DNA sequence. *Genome research*. 2005, 21, 2167-2180.

See Also

See `make_kmer_index`

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNAkmer(x)
```

`extrDNAPseDNC` *The Pseudo Dinucleotide Composition Descriptor*

Description

The Pseudo Dinucleotide Composition Descriptor

Usage

```
extrDNAPseDNC(x, lambda = 3, w = 0.05, normalize = FALSE,
  customprops = NULL)
```

Arguments

`x`	the input data, which should be a list or file type.
`lambda`	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest sequence in the dataset). It represents the highest counted rank (or tier) of the correlation along a DNA sequence. Its default value is 3.
`w`	the weight factor ranged from 0 to 1. Its default value is 0.05.
`normalize`	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
`customprops`	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.

Details

This function calculates the pseudo dinucleotide composition Descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Chen W, Feng P M, Lin H, et al. iRSpot-PseDNC: identify recombination spots with pseudo dinucleotide composition. *Nucleic acids research*, 2013: gks1450.

See Also

See extrDNAPseKNC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNAPseDNC(x)
```

---

extrDNAPseKNC *The Pseudo K-tupler Composition Descriptor*

Description

The Pseudo K-tupler Composition Descriptor

Usage

```
extrDNAPseKNC(x, lambda = 1, k = 3, normalize = FALSE, w = 0.5,
  customprops = NULL)
```

Arguments

x	the input data, which should be a list or file type.
lambda	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest sequence in the dataset). It represents the highest counted rank (or tier) of the correlation along a DNA sequence. Its default value is 3.
k	an integer larger than 0 represents the k-tuple. Its default value is 3.
normalize	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
w	the weight factor ranged from 0 to 1. Its default value is 0.05.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.

Details

This function calculates the pseudo k-tupler composition Descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Guo S H, Deng E Z, Xu L Q, et al. iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition. *Bioinformatics*, 2014: btu083.

See Also

See extrDNAPseDNC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNAPseKNC(x)
```

---

extrDNATAC *The Trinucleotide-based Auto Covariance Descriptor*

Description

The Trinucleotide-based Auto Covariance Descriptor

Usage

```
extrDNATAC(x, index = c("Dnase I", "Nucleosome"), nlag = 2,
  normaliztion = FALSE, customprops = NULL, allprop = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 12 different physicochemical indices (Table 2), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.
normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
allprop	all the 12 physicochemical indices will be employed to generate the feature vector. Its default value is False.

Details

This function calculates the trinucleotide-based auto covariance Descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See extrDNATCC and extrDNATACC

Examples

```
x = x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNATAC(x)
```

---

extrDNATACC *The Trinucleotide-based Auto-cross Covariance Descriptor*

Description

The Trinucleotide-based Auto-cross Covariance Descriptor

Usage

```
extrDNATACC(x, index = c("Dnase I", "Nucleosome"), nlag = 2,
  normaliztion = FALSE, customprops = NULL, allprop = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 12 different physicochemical indices (Table 2), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.
normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
allprop	all the 12 physicochemical indices will be employed to generate the feature vector. Its default value is False.

Details

This function calculates the trinucleotide-based auto-cross covariance descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See extrDNATAC and extrDNATCC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNATACC(x)
```

---

extrDNATCC *The Trinucleotide-based Cross Covariance Descriptor*

Description

The Trinucleotide-based Cross Covariance Descriptor

Usage

```
extrDNATCC(x, index = c("Dnase I", "Nucleosome"), nlag = 2,
  customprops = NULL, normaliztion = FALSE)
```

Arguments

x	the input data, which should be a list or file type.
index	the physicochemical indices, it should be a list and there are 12 different physicochemical indices (Table 2), which the users can choose.
nlag	an integer larger than or equal to 0 and less than or equal to L-2 (L means the length of the shortest DNA sequence in the dataset). It represents the distance between two dinucleotides.
customprops	the users can use their own indices to generate the feature vector. It should be a dict, the key is dinucleotide (string), and its corresponding value is a list type.
normaliztion	with this option, the final feature vector will be normalized based on the total occurrences of all kmers. Therefore, the elements in the feature vectors represent the frequencies of kmers. The default value of this parameter is False.

Details

This function calculates the trinucleotide-based cross covariance Descriptor

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See extrDNATAC and extrDNATACC

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
extrDNATCC(x)
```

---

extrDrugAIO *Calculates All the Molecular Descriptors in the BioMedR Package at Once*

Description

Calculates All the Molecular Descriptors in the BioMedR Package at Once

Usage

```
extrDrugAIO(molecules, silent = TRUE, warn = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.
warn	Logical. Whether the warning about some descriptors need the 3D coordinates should be shown or not after the calculation, default is TRUE.

Details

This function calculates all the molecular descriptors in the BioMedR package at once.

Value

A data frame, each row represents one of the molecules, each column represents one descriptor. Currently, this function returns total 293 descriptors composed of 48 descriptor types.

Note

Note that we need 3-D coordinates of the molecules to calculate some of the descriptors, if not provided, these descriptors values will be NA.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
# Load 20 small molecules that have 3D coordinates
sdf = system.file('sysdata/test.sdf', package = 'BioMedR')
mol = readMolFromSDF(sdf)
dat = extrDrugAIO(mol, warn = FALSE)
```

---

extrDrugAP *Calculate the Atom Pair Fingerprints*

Description

Calculate the Atom Pair Fingerprints

Usage

```
extrDrugAP(x, descnames = 1024)
```

Arguments

x	Object of classe APset or list of vectors
descnames	Descriptor set to consider for fingerprint encoding. If a single value from 1-4096 is provided then the function uses the corresponding number of the most frequent atom pairs stored in the apfp data set provided by the package. Alternatively, one can provide here any custom atom pair selection in form of a character vector.

Details

Generates fingerprints from descriptor vectors such as atom pairs stored in APset or list containers. The obtained fingerprints can be used for structure similarity comparisons, searching and clustering. Due to their compact size, computations on fingerprints are often more time and memory efficient than on their much more complex atom pair counterparts.

Value

matrix or character vectors

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>,

Examples

```
data(sdfbcl)

apbcl <- convSDFtoAP(sdfbcl)
mol <- extrDrugAP(x = apbcl, descnames = 1024)
```

`extrDrugBCUT` *BCUT – Eigenvalue Based Descriptor*

Description

BCUT – Eigenvalue Based Descriptor

Usage

```
extrDrugBCUT(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Eigenvalue based descriptor noted for its utility in chemical diversity. Described by Pearlman et al. The descriptor is based on a weighted version of the Burden matrix which takes into account both the connectivity as well as atomic properties of a molecule. The weights are a variety of atom properties placed along the diagonal of the Burden matrix. Currently three weighting schemes are employed:

- Atomic Weight
- Partial Charge (Gasteiger Marsilli)
- Polarizability (Kang et al.)

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 6 columns:

- `BCUTw-1l, BCUTw-2l ...` - n high lowest atom weighted BCUTS
- `BCUTw-1h, BCUTw-2h ...` - n low highest atom weighted BCUTS
- `BCUTc-1l, BCUTc-2l ...` - n high lowest partial charge weighted BCUTS
- `BCUTc-1h, BCUTc-2h ...` - n low highest partial charge weighted BCUTS
- `BCUTp-1l, BCUTp-2l ...` - n high lowest polarizability weighted BCUTS
- `BCUTp-1h, BCUTp-2h ...` - n low highest polarizability weighted BCUTS

Note

By default, the descriptor will return the highest and lowest eigenvalues for the three classes of descriptor in a single ArrayList (in the order shown above). However it is also possible to supply a parameter list indicating how many of the highest and lowest eigenvalues (for each class of descriptor) are required. The descriptor works with the hydrogen depleted molecule.

A side effect of specifying the number of highest and lowest eigenvalues is that it is possible to get two copies of all the eigenvalues. That is, if a molecule has 5 heavy atoms, then specifying the 5 highest eigenvalues returns all of them, and specifying the 5 lowest eigenvalues returns all of them, resulting in two copies of all the eigenvalues.

Note that it is possible to specify an arbitrarily large number of eigenvalues to be returned. However if the number (i.e., nhigh or nlow) is larger than the number of heavy atoms, the remaining eignevalues will be NaN.

Given the above description, if the aim is to gt all the eigenvalues for a molecule, you should set nlow to 0 and specify the number of heavy atoms (or some large number) for nhigh (or vice versa).

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Pearlman, R.S. and Smith, K.M., Metric Validation and the Receptor-Relevant Subspace Concept, J. Chem. Inf. Comput. Sci., 1999, 39:28-35.

Burden, F.R., Molecular identification number for substructure searches, J. Chem. Inf. Comput. Sci., 1989, 29:225-227.

Burden, F.R., Chemically Intuitive Molecular Index, Quant. Struct. -Act. Relat., 1997, 16:309-314

Kang, Y.K. and Jhon, M.S., Additivity of Atomic Static Polarizabilities and Dispersion Coefficients, Theoretica Chimica Acta, 1982, 61:41-48

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugBCUT(mol)
head(dat)
```

---

extrDrugCPSA *A Variety of Descriptors Combining Surface Area and Partial Charge Information*

Description

A Variety of Descriptors Combining Surface Area and Partial Charge Information

Usage

```
extrDrugCPSA(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates 29 Charged Partial Surface Area (CPSA) descriptors. The CPSA's were developed by Stanton et al.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 29 columns:

- PPSA.1 - partial positive surface area – sum of surface area on positive parts of molecule
- PPSA.2 - partial positive surface area * total positive charge on the molecule
- PPSA.3 - charge weighted partial positive surface area
- PNSA.1 - partial negative surface area – sum of surface area on negative parts of molecule
- PNSA.2 - partial negative surface area * total negative charge on the molecule
- PNSA.3 - charge weighted partial negative surface area
- DPSA.1 - difference of PPSA.1 and PNSA.1
- DPSA.2 - difference of FPSA.2 and PNSA.2
- DPSA.3 - difference of PPSA.3 and PNSA.3
- FPSA.1 - PPSA.1 / total molecular surface area
- FFSA.2 - PPSA.2 / total molecular surface area
- FPSA.3 - PPSA.3 / total molecular surface area
- FNSA.1 - PNSA.1 / total molecular surface area
- FNSA.2 - PNSA.2 / total molecular surface area
- FNSA.3 - PNSA.3 / total molecular surface area
- WPSA.1 - PPSA.1 * total molecular surface area / 1000
- WPSA.2 - PPSA.2 * total molecular surface area /1000
- WPSA.3 - PPSA.3 * total molecular surface area / 1000
- WNSA.1 - PNSA.1 * total molecular surface area /1000
- WNSA.2 - PNSA.2 * total molecular surface area / 1000
- WNSA.3 - PNSA.3 * total molecular surface area / 1000
- RPCG - relative positive charge – most positive charge / total positive charge
- RNCG - relative negative charge – most negative charge / total negative charge
- RPCS - relative positive charge surface area – most positive surface area * RPCG
- RNCS - relative negative charge surface area – most negative surface area * RNCG
- THSA - sum of solvent accessible surface areas of atoms with absolute value of partial charges less than 0.2
- TPSA - sum of solvent accessible surface areas of atoms with absolute value of partial charges greater than or equal 0.2
- RHSA - THSA / total molecular surface area
- RPSA - TPSA / total molecular surface area

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Stanton, D.T. and Jurs, P.C. , Development and Use of Charged Partial Surface Area Structural Descriptors in Computer Assissted Quantitative Structure Property Relationship Studies, Analytical Chemistry, 1990, 62:2323.2329.

Examples

```
sdf = system.file('sysdata/test.sdf', package = 'BioMedR')
mol = readMolFromSDF(sdf)
dat = extrDrugCPSA(mol)
head(dat)
```

---

extrDrugEstate *Calculate the E-State Molecular Fingerprints (in Compact Format)*

Description

Calculate the E-State Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugEstate(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

79 bit fingerprints corresponding to the E-State atom types described by Hall and Kier.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugEstateComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugEstate(mol)
head(fp)
```

---

extrDrugEstateComplete

Calculate the E-State Molecular Fingerprints (in Complete Format)

Description

Calculate the E-State Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugEstateComplete(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

79 bit fingerprints corresponding to the E-State atom types described by Hall and Kier.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugEstate

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugEstateComplete(mol)
dim(fp)
```

`extrDrugExtended` *Calculate the Extended Molecular Fingerprints (in Compact Format)*

Description

Calculate the Extended Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugExtended(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the extended molecular fingerprints. Considers paths of a given length, similar to the standard type, but takes rings and atomic properties into account into account. This is hashed fingerprints, with a default length of 1024.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugExtendedComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugExtended(mol)
head(fp)
```

`extrDrugExtendedComplete` *Calculate the Extended Molecular Fingerprints (in Complete Format)*

Description

Calculate the Extended Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugExtendedComplete(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the extended molecular fingerprints. Considers paths of a given length, similar to the standard type, but takes rings and atomic properties into account into account. This is hashed fingerprints, with a default length of 1024.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugExtended

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugExtendedComplete(mol)
dim(fp)
```

`extrDrugGraph` *Calculate the Graph Molecular Fingerprints (in Compact Format)*

Description

Calculate the Graph Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugGraph(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the graph molecular fingerprints. Similar to the standard type by simply considers connectivity. This is hashed fingerprints, with a default length of 1024.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugGraphComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugGraph(mol)
head(fp)
```

`extrDrugGraphComplete` *Calculate the Graph Molecular Fingerprints (in Complete Format)*

Description

Calculate the Graph Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugGraphComplete(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the graph molecular fingerprints. Similar to the standard type by simply considers connectivity. This is hashed fingerprints, with a default length of 1024.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugGraph

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugGraphComplete(mol)
dim(fp)
```

`extrDrugHybridization` *Calculate the Hybridization Molecular Fingerprints (in Compact Format)*

Description

Calculate the Hybridization Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugHybridization(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the hybridization molecular fingerprints. Similar to the standard type, but only consider hybridization state. This is hashed fingerprints, with a default length of 1024.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugHybridizationComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugHybridization(mol)
head(fp)
```

`extrDrugHybridizationComplete` *Calculate the Hybridization Molecular Fingerprints (in Complete Format)*

Description

Calculate the Hybridization Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugHybridizationComplete(molecules, depth = 6, size = 1024,
  silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the hybridization molecular fingerprints. Similar to the standard type, but only consider hybridization state. This is hashed fingerprints, with a default length of 1024.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugHybridization

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugHybridizationComplete(mol)
dim(fp)
```

`extrDrugHybridizationRatio` *Descriptor that Characterizing Molecular Complexity in Terms of Carbon Hybridization States*

Description

Descriptor that Characterizing Molecular Complexity in Terms of Carbon Hybridization States

Usage

```
extrDrugHybridizationRatio(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

This descriptor calculates the fraction of sp3 carbons to sp2 carbons. Note that it only considers carbon atoms and rather than use a simple ratio it reports the value of Nsp3/(Nsp3 + Nsp2). The original form of the descriptor (i.e., simple ratio) has been used to characterize molecular complexity, especially in the are of natural products, which usually have a high value of the sp3 to sp2 ratio.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `HybRatio`.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugHybridizationRatio(mol)
head(dat)
```

extrDrugIPMolecularLearning

Calculates the Descriptor that Evaluates the Ionization Potential

Description

Calculates the Descriptor that Evaluates the Ionization Potential

Usage

```
extrDrugIPMolecularLearning(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the ionization potential of a molecule. The descriptor assumes that explicit hydrogens have been added to the molecules.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named MolIP.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugIPMolecularLearning(mol)
head(dat)
```

extrDrugKappaShapeIndices

Descriptor that Calculates Kier and Hall Kappa Molecular Shape Indices

Description

Descriptor that Calculates Kier and Hall Kappa Molecular Shape Indices

Usage

```
extrDrugKappaShapeIndices(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Kier and Hall Kappa molecular shape indices compare the molecular graph with minimal and maximal molecular graphs; see http://www.chemcomp.com/Journal_of_CCG/Features/descr.htm#KH for details: "they are intended to capture different aspects of molecular shape. Note that hydrogens are ignored. In the following description, n denotes the number of atoms in the hydrogen suppressed graph, m is the number of bonds in the hydrogen suppressed graph. Also, let p2 denote the number of paths of length 2 and let p3 denote the number of paths of length 3".

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 3 columns named Kier1, Kier2 and Kier3:

- Kier1 - First kappa shape index
- Kier2 - Second kappa shape index
- Kier3 - Third kappa shape index

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugKappaShapeIndices(mol)
head(dat)
```

---

extrDrugKierHallSmarts

Descriptor that Counts the Number of Occurrences of the E-State Fragments

---

Description

Descriptor that Counts the Number of Occurrences of the E-State Fragments

Usage

```
extrDrugKierHallSmarts(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

A fragment count descriptor that uses e-state fragments. Traditionally the e-state descriptors identify the relevant fragments and then evaluate the actual e-state value. However it has been shown in Butina et al. that simply using the counts of the e-state fragments can lead to QSAR models that exhibit similar performance to those built using the actual e-state indices.

Atom typing and aromaticity perception should be performed prior to calling this descriptor. The atom type definitions are taken from Hall et al. The SMARTS definitions were obtained from RDKit.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 79 columns:

ID	Name	Pattern
0	khs.sLi	[LiD1]-*
1	khs.ssBe	[BeD2](-*)-*
2	khs.ssssBe	[BeD4](-*)(-*)(-*)-*
3	khs.ssBH	[BD2H](-*)-*
4	khs.sssB	[BD3](-*)(-*)-*
5	khs.ssssB	[BD4](-*)(-*)(-*)-*
6	khs.sCH3	[CD1H3]-*
7	khs.dCH2	[CD1H2]=*
8	khs.ssCH2	[CD2H2](-*)-*
9	khs.tCH	[CD1H]#*
10	khs.dsCH	[CD2H](=*)-*
11	khs.aaCH	[C,c;D2H](:*):*
12	khs.sssCH	[CD3H](-*)(-*)-*
13	khs.ddC	[CD2H0](=*)=*
14	khs.tsC	[CD2H0](#*)-*
15	khs.dssC	[CD3H0](=*)(-*)-*
16	khs.aasC	[C,c;D3H0](:*)(:*)-*
17	khs.aaaC	[C,c;D3H0](:*)(:*):*
18	khs.ssssC	[CD4H0](-*)(-*)(-*)-*
19	khs.sNH3	[ND1H3]-*
20	khs.sNH2	[ND1H2]-*
21	khs.ssNH2	[ND2H2](-*)-*
22	khs.dNH	[ND1H]=*
23	khs.ssNH	[ND2H](-*)-*
24	khs.aaNH	[N,nD2H](:*):*
25	khs.tN	[ND1H0]#*
26	khs.sssNH	[ND3H](-*)(-*)-*
27	khs.dsN	[ND2H0](=*)-*
28	khs.aaN	[N,nD2H0](:*):*
29	khs.sssN	[ND3H0](-*)(-*)-*
30	khs.ddsN	[ND3H0](~[OD1H0])(~[OD1H0])-,:*
31	khs.aasN	[N,nD3H0](:*)(:*)-,:*
32	khs.ssssN	[ND4H0](-*)(-*)(-*)-*
33	khs.sOH	[OD1H]-*
34	khs.dO	[OD1H0]=*
35	khs.ssO	[OD2H0](-*)-*
36	khs.aaO	[O,oD2H0](:*):*

37	khs.sF	[FD1]-*
38	khs.sSiH3	[SiD1H3]-*
39	khs.ssSiH2	[SiD2H2](-*)-*
40	khs.sssSiH	[SiD3H1](-*)(-*)-*
41	khs.ssssSi	[SiD4H0](-*)(-*)(-*)-*
42	khs.sPH2	[PD1H2]-*
43	khs.ssPH	[PD2H1](-*)-*
44	khs.sssP	[PD3H0](-*)(-*)-*
45	khs.dsssP	[PD4H0](=*)(-*)(-*)-*
46	khs.sssssP	[PD5H0](-*)(-*)(-*)(-*)-*
47	khs.sSH	[SD1H1]-*
48	khs.dS	[SD1H0]=*
49	khs.ssS	[SD2H0](-*)-*
50	khs.aaS	[S,sD2H0](:*):*
51	khs.dssS	[SD3H0](=*)(-*)-*
52	khs.ddssS	[SD4H0](~[OD1H0])(~[OD1H0])(-*)-*
53	khs.sCl	[ClD1]-*
54	khs.sGeH3	[GeD1H3](-*)
55	khs.ssGeH2	[GeD2H2](-*)-*
56	khs.sssGeH	[GeD3H1](-*)(-*)-*
57	khs.ssssGe	[GeD4H0](-*)(-*)(-*)-*
58	khs.sAsH2	[AsD1H2]-*
59	khs.ssAsH	[AsD2H1](-*)-*
60	khs.sssAs	[AsD3H0](-*)(-*)-*
61	khs.sssdAs	[AsD4H0](=*)(-*)(-*)-*
62	khs.sssssAs	[AsD5H0](-*)(-*)(-*)(-*)-*
63	khs.sSeH	[SeD1H1]-*
64	khs.dSe	[SeD1H0]=*
65	khs.ssSe	[SeD2H0](-*)-*
66	khs.aaSe	[SeD2H0](:*):*
67	khs.dssSe	[SeD3H0](=*)(-*)-*
68	khs.ddssSe	[SeD4H0](=*)(=*)(-*)-*
69	khs.sBr	[BrD1]-*
70	khs.sSnH3	[SnD1H3]-*
71	khs.ssSnH2	[SnD2H2](-*)-*
72	khs.sssSnH	[SnD3H1](-*)(-*)-*
73	khs.ssssSn	[SnD4H0](-*)(-*)(-*)-*
74	khs.sI	[ID1]-*
75	khs.sPbH3	[PbD1H3]-*
76	khs.ssPbH2	[PbD2H2](-*)-*
77	khs.sssPbH	[PbD3H1](-*)(-*)-*
78	khs.ssssPb	[PbD4H0](-*)(-*)(-*)-*

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Butina, D. , Performance of Kier-Hall E-state Descriptors in Quantitative Structure Activity Relationship (QSAR) Studies of Multifunctional Molecules, Molecules, 2004, 9:1004-1009.

Hall, L.H. and Kier, L.B. , Electrotopological State Indices for Atom Types: A Novel Combination

of Electronic, Topological, and Valence State Information, Journal of Chemical Information and Computer Science, 1995, 35:1039-1045.

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugKierHallSmarts(mol)
head(dat)
```

---

extrDrugKR — *Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Compact Format)*

Description

Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugKR(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 4860 bit fingerprint defined by Klekota and Roth.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugKRComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugKR(mol)
head(fp)
```

`extrDrugKRComplete` *Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Complete Format)*

Description

Calculate the KR (Klekota and Roth) Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugKRComplete(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the 4860 bit fingerprint defined by Klekota and Roth.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugKR

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugKRComplete(mol)
dim(fp)
```

`extrDrugMACCS` *Calculate the MACCS Molecular Fingerprints (in Compact Format)*

Description

Calculate the MACCS Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugMACCS(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

The popular 166 bit MACCS keys described by MDL.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugMACCSComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugMACCS(mol)
head(fp)
```

`extrDrugMACCSComplete` *Calculate the MACCS Molecular Fingerprints (in Complete Format)*

Description

Calculate the MACCS Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugMACCSComplete(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

The popular 166 bit MACCS keys described by MDL.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugMACCS

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugMACCSComplete(mol)
dim(fp)
```

`extrDrugMannholdLogP` *Descriptor that Calculates the LogP Based on a Simple Equation Using the Number of Carbons and Hetero Atoms*

Description

Descriptor that Calculates the LogP Based on a Simple Equation Using the Number of Carbons and Hetero Atoms

Usage

```
extrDrugMannholdLogP(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

This descriptor calculates the LogP based on a simple equation using the number of carbons and hetero atoms. The implemented equation was proposed in Mannhold et al.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns one column named `MLogP`.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Mannhold, R., Poda, G. I., Ostermann, C., & Tetko, I. V. (2009). Calculation of molecular lipophilicity: State-of-the-art and comparison of log P methods on more than 96,000 compounds. Journal of pharmaceutical sciences, 98(3), 861-893.

Examples

```
smi = system.file('vignettedata/FDAMDD.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugMannholdLogP(mol)
head(dat)
```

extrDrugOBFP2 *Calculate the FP2 Molecular Fingerprints*

Description

Calculate the FP2 Molecular Fingerprints

Usage

```
extrDrugOBFP2(molecules, type = c("smile", "sdf"))
```

Arguments

molecules	R character string object containing the molecules. See the example section for details.
type	'smile' or 'sdf'.

Details

Calculate the 1024 bit FP2 fingerprints provided by OpenBabel.

Value

A matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
mol1 = 'C1CCC1CC(CN(C)(C))CC(=O)CC'  # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1ccccc1Cc1ccccc1',
         'C1CCC1CC(CN(C)(C))CC(=O)CC')  # multiple SMILEs in a vector

smifp0 = extrDrugOBFP2(mol1, type = 'smile')
smifp1 = extrDrugOBFP2(mol2, type = 'smile')
```

extrDrugOBFP3 *Calculate the FP3 Molecular Fingerprints*

Description

Calculate the FP3 Molecular Fingerprints

Usage

```
extrDrugOBFP3(molecules, type = c("smile", "sdf"))
```

Arguments

molecules	R character string object containing the molecules. See the example section for details.
type	'smile' or 'sdf'.

Details

Calculate the 64 bit FP3 fingerprints provided by OpenBabel.

Value

A matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
mol1 = 'C1CCC1CC(CN(C)(C))CC(=O)CC'  # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1ccccc1Cc1ccccc1',
         'C1CCC1CC(CN(C)(C))CC(=O)CC')  # multiple SMILEs in a vector

smifp0 = extrDrugOBFP3(mol1, type = 'smile')
smifp1 = extrDrugOBFP3(mol2, type = 'smile')
```

---

extrDrugOBFP4 *Calculate the FP4 Molecular Fingerprints*

Description

Calculate the FP4 Molecular Fingerprints

Usage

```
extrDrugOBFP4(molecules, type = c("smile", "sdf"))
```

Arguments

molecules	R character string object containing the molecules. See the example section for details.
type	'smile' or 'sdf'.

Details

Calculate the 512 bit FP4 fingerprints provided by OpenBabel.

Value

A matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
mol1 = 'C1CCC1CC(CN(C)(C))CC(=O)CC'  # one molecule SMILE in a vector
mol2 = c('CCC', 'CCN', 'CCN(C)(C)', 'c1ccccc1Cc1ccccc1',
         'C1CCC1CC(CN(C)(C))CC(=O)CC')  # multiple SMILEs in a vector

smifp0 = extrDrugOBFP4(mol1, type = 'smile')
smifp1 = extrDrugOBFP4(mol2, type = 'smile')
```

---

extrDrugPubChem *Calculate the PubChem Molecular Fingerprints (in Compact Format)*

Description

Calculate the PubChem Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugPubChem(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the 881 bit fingerprints defined by PubChem.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugPubChemComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugPubChem(mol)
head(fp)
```

`extrDrugPubChemComplete` *Calculate the PubChem Molecular Fingerprints (in Complete Format)*

Description

Calculate the PubChem Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugPubChemComplete(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the 881 bit fingerprints defined by PubChem.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugPubChem

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugPubChemComplete(mol)
dim(fp)
```

`extrDrugShortestPath` *Calculate the Shortest Path Molecular Fingerprints (in Compact Format)*

Description

Calculate the Shortest Path Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugShortestPath(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the fingerprint based on the shortest paths between pairs of atoms and takes into account ring systems, charges etc.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugShortestPathComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugShortestPath(mol)
head(fp)
```

`extrDrugShortestPathComplete` *Calculate the Shortest Path Molecular Fingerprints (in Complete Format)*

Description

Calculate the Shortest Path Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugShortestPathComplete(molecules, depth = 6, size = 1024,
  silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the fingerprint based on the shortest paths between pairs of atoms and takes into account ring systems, charges etc.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugShortestPath

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugShortestPathComplete(mol)
dim(fp)
```

extrDrugStandard *Calculate the Standard Molecular Fingerprints (in Compact Format)*

Description

Calculate the Standard Molecular Fingerprints (in Compact Format)

Usage

```
extrDrugStandard(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
depth	The search depth. Default is 6.
size	The length of the fingerprint bit string. Default is 1024.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculate the standard molecular fingerprints. Considers paths of a given length. This is hashed fingerprints, with a default length of 1024.

Value

A list, each component represents one of the molecules, each element in the component represents the index of which element in the fingerprint is 1. Each component's name is the length of the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugStandardComplete

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugStandard(mol)
head(fp)
```

`extrDrugStandardComplete` *Calculate the Standard Molecular Fingerprints (in Complete Format)*

Description

Calculate the Standard Molecular Fingerprints (in Complete Format)

Usage

```
extrDrugStandardComplete(molecules, depth = 6, size = 1024, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`depth`	The search depth. Default is `6`.
`size`	The length of the fingerprint bit string. Default is `1024`.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Calculate the standard molecular fingerprints. Considers paths of a given length. This is hashed fingerprints, with a default length of 1024.

Value

An integer vector or a matrix. Each row represents one molecule, the columns represent the fingerprints.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

extrDrugStandard

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
fp  = extrDrugStandardComplete(mol)
dim(fp)
```

`extrDrugWHIM` *Calculate Holistic Descriptors Described by Todeschini et al.*

Description

Calculate Holistic Descriptors Described by Todeschini et al.

Usage

```
extrDrugWHIM(molecules, silent = TRUE)
```

Arguments

`molecules`	Parsed molucule object.
`silent`	Logical. Whether the calculating process should be shown or not, default is `TRUE`.

Details

Holistic descriptors described by Todeschini et al, the descriptors are based on a number of atom weightings. There are six different possible weightings:

- unit weights
- atomic masses
- van der Waals volumes
- Mulliken atomic electronegativites
- atomic polarizabilities
- E-state values described by Kier and Hall

Currently weighting schemes 1, 2, 3, 4 and 5 are implemented. The weight values are taken from Todeschini et al. and as a result 19 elements are considered. For each weighting scheme we can obtain

- 11 directional WHIM descriptors (lambda1 .. 3, nu1 .. 2, gamma1 .. 3, eta1 .. 3)
- 6 non-directional WHIM descriptors (T, A, V, K, G, D)

Though Todeschini et al. mentions that for planar molecules only 8 directional WHIM descriptors are required the current code will return all 11.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 17 columns:

- Wlambda1
- Wlambda2
- wlambda3
- Wnu1
- Wnu2

- Wgamma1
- Wgamma2
- Wgamma3
- Weta1
- Weta2
- Weta3
- WT
- WA
- WV
- WK
- WG
- WD

Each name will have a suffix of the form `.X` where `X` indicates the weighting scheme used. Possible values of `X` are

- unity
- mass
- volume
- eneg
- polar

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Todeschini, R. and Gramatica, P., New 3D Molecular Descriptors: The WHIM theory and QAR Applications, Persepectives in Drug Discovery and Design, 1998, ?:355-380.

Examples

```
sdf = system.file('sysdata/test.sdf', package = 'BioMedR')
mol = readMolFromSDF(sdf)
dat = extrDrugWHIM(mol)
head(dat)
```

`extrPCMBLOSUM` *Generalized BLOSUM and PAM Matrix-Derived Descriptors*

Description

Generalized BLOSUM and PAM Matrix-Derived Descriptors

Usage

```
extrPCMBLOSUM(x, submat = "AABLOSUM62", k, lag, scale = TRUE,
  silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`submat`	Substitution matrix for the 20 amino acids. Should be one of `AABLOSUM45`, `AABLOSUM50`, `AABLOSUM62`, `AABLOSUM80`, `AABLOSUM100`, `AAPAM30`, `AAPAM40`, `AAPAM70`, `AAPAM120`, `AAPAM250`. Default is `'AABLOSUM62'`.
`k`	Integer. The number of selected scales (i.e. the first `k` scales) derived by the substitution matrix. This could be selected according to the printed relative importance values.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the substitution matrix (`submat`) before doing eigen decomposition? Default is `TRUE`.
`silent`	Logical. Whether we print the relative importance of each scales (diagnal value of the eigen decomposition result matrix B) or not. Default is `TRUE`.

Details

This function calculates the generalized BLOSUM matrix-derived descriptors. For users' convenience, `BioMedR` provides the BLOSUM45, BLOSUM50, BLOSUM62, BLOSUM80, BLOSUM100, PAM30, PAM40, PAM70, PAM120, and PAM250 matrices for the 20 amino acids to select.

Value

A length `lag * p^2` named vector, `p` is the number of scales selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Georgiev, A. G. (2009). Interpretable numerical descriptors of amino acid space. Journal of Computational Biology, 16(5), 703–723.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
blosum = extrPCMBLOSUM(x, submat = 'AABLOSUM62', k = 5, lag = 7, scale = TRUE, silent = FALSE)
```

`extrPCMDescScales` *Scales-Based Descriptors with 20+ classes of Molecular Descriptors*

Description

Scales-Based Descriptors with 20+ classes of Molecular Descriptors

Usage

```
extrPCMDescScales(x, propmat, index = NULL, pc, lag, scale = TRUE,
  silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`propmat`	The matrix containing the descriptor set for the amino acids, which could be chosen from `AAMOE2D`, `AAMOE3D`, `AACPSA`, `AADescAll`, `AA2DACOR`, `AA3DMoRSE`, `AAACF`, `AABurden`, `AAConn`, `AAConst`, `AAEdgeAdj`, `AAEigIdx`, `AAFGC`, `AAGeom`, `AAGETAWAY`, `AAInfo`, `AAMolProp`, `AARandic`, `AARDF`, `AATopo`, `AATopoChg`, `AAWalk`, `AAWHIM`.
`index`	Integer vector or character vector. Specify which molecular descriptors to select from one of these deseriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set. Default is `NULL`, means selecting all the molecular descriptors in this descriptor set.
`pc`	Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix (`propmat`) before doing MDS? Default is `TRUE`.
`silent`	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is `TRUE`.

Details

This function calculates the scales-based descriptors with molecular descriptors sets calculated by Dragon, Discovery Studio and MOE. Users could specify which molecular descriptors to select from one of these deseriptor sets by specify the numerical or character index of the molecular descriptors in the descriptor set.

Value

A length `lag * p^2` named vector, `p` is the number of scales selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `extrPCMScales` for generalized AA-descriptor based scales descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
descscales = extrPCMDescScales(x, propmat = 'AATopo', index = c(37:41, 43:47),
                                  pc = 5, lag = 7, silent = FALSE)
```

---

extrPCMFAScales *Generalized Scales-Based Descriptors derived by Factor Analysis*

Description

Generalized Scales-Based Descriptors derived by Factor Analysis

Usage

```
extrPCMFAScales(x, propmat, factors, scores = "regression", lag,
  scale = TRUE, silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence.
propmat	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
factors	Integer. The number of factors to be fitted. Must be no greater than the number of AA properties provided.
scores	Type of scores to produce. The default is "regression", which gives Thompson's scores, "Bartlett" given Bartlett's weighted least-squares scores.
lag	The lag parameter. Must be less than the amino acids number in the protein sequence.
scale	Logical. Should we auto-scale the property matrix (propmat) before doing Factor Analysis? Default is TRUE.
silent	Logical. Whether we print the SS loadings, proportion of variance and the cumulative proportion of the selected factors or not. Default is TRUE.

Details

This function calculates the generalized scales-based descriptors derived by Factor Analysis (FA). Users could provide customized amino acid property matrices.

Value

A length lag * p^2 named vector, p is the number of scales (factors) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Atchley, W. R., Zhao, J., Fernandes, A. D., & Druke, T. (2005). Solving the protein sequence metric problem. Proceedings of the National Academy of Sciences of the United States of America, 102(18), 6395-6400.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)]  # select a set of topological descriptors
fa = extrPCMFAScales(x, propmat = tprops, factors = 5, lag = 7, silent = FALSE)
```

`extrPCMMDSScales` *Generalized Scales-Based Descriptors derived by Multidimensional Scaling*

Description

Generalized Scales-Based Descriptors derived by Multidimensional Scaling

Usage

```
extrPCMMDSScales(x, propmat, k, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`propmat`	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
`k`	Integer. The maximum dimension of the space which the data are to be represented in. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix (`propmat`) before doing MDS? Default is `TRUE`.
`silent`	Logical. Whether we print the `k` eigenvalues computed during the scaling process or not. Default is `TRUE`.

Details

This function calculates the generalized scales-based descriptors derived by Multidimensional Scaling (MDS). Users could provide customized amino acid property matrices.

Value

A length `lag * p^2` named vector, `p` is the number of scales (dimensionality) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Venkatarajan, M. S., & Braun, W. (2001). New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical-chemical properties. Molecular modeling annual, 7(12), 445–453.

See Also

See extrPCMScales for generalized scales-based descriptors derived by Principal Components Analysis.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
data(AATopo)
tprops = AATopo[, c(37:41, 43:47)]  # select a set of topological descriptors
mds = extrPCMMDSScales(x, propmat = tprops, k = 5, lag = 7, silent = FALSE)
```

---

extrPCMPropScales *Generalized AA-Properties Based Scales Descriptors*

Description

Generalized AA-Properties Based Scales Descriptors

Usage

```
extrPCMPropScales(x, index = NULL, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

x	A character vector, as the input protein sequence.
index	Integer vector or character vector. Specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database. Default is NULL, means selecting all the AA properties in the AAindex database.
pc	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
lag	The lag parameter. Must be less than the amino acids.
scale	Logical. Should we auto-scale the property matrix before PCA? Default is TRUE.
silent	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is TRUE.

Details

This function calculates the generalized amino acid properties based scales descriptors. Users could specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `extrPCMScales` for generalized scales-based descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
propscales = extrPCMPropScales(x, index = c(160:165, 258:296), pc = 5, lag = 7, silent = FALSE)
```

`extrPCMScaleGap` *Scales-Based Descriptors derived by Principal Components Analysis (with Gap Support)*

Description

Scales-Based Descriptors derived by Principal Components Analysis (with Gap Support)

Usage

```
extrPCMScaleGap(x, propmat, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence. Use '-' to represent gaps in the sequence.
`propmat`	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
`pc`	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix (`propmat`) before PCA? Default is `TRUE`.
`silent`	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is `TRUE`.

Details

This function calculates scales-based descriptors derived by Principal Components Analysis (PCA), with gap support. Users could provide customized amino acid property matrices. This function implements the core computation procedure needed for the scales-based descriptors derived by AA-Properties (AAindex) and scales-based descriptors derived by 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.) in the BioMedR package.

Value

A length `lag * p^2` named vector, `p` is the number of scales (principal components) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

See Also

See `extrProtFPGap` for amino acid property based scales descriptors (protein fingerprint) with gap support.

Examples

```
# amino acid sequence with gaps
x = readFASTA(system.file('protseq/align.fasta', package = 'BioMedR'))$`IXI_235`
data(AAindex)
AAidxmat = t(na.omit(as.matrix(AAindex[, 7:26])))
scales = extrPCMScaleGap(x, propmat = AAidxmat, pc = 5, lag = 7, silent = FALSE)
```

`extrPCMScales` *Generalized Scales-Based Descriptors derived by Principal Components Analysis*

Description

Generalized Scales-Based Descriptors derived by Principal Components Analysis

Usage

```
extrPCMScales(x, propmat, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence.
`propmat`	A matrix containing the properties for the amino acids. Each row represent one amino acid type, each column represents one property. Note that the one-letter row names must be provided for we need them to seek the properties for each AA type.
`pc`	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix (`propmat`) before PCA? Default is `TRUE`.
`silent`	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is `TRUE`.

Details

This function calculates the generalized scales-based descriptors derived by Principal Components Analysis (PCA). Users could provide customized amino acid property matrices. This function implements the core computation procedure needed for the generalized scales-based descriptors derived by AA-Properties (AAindex) and generalized scales-based descriptors derived by 20+ classes of 2D and 3D molecular descriptors (Topological, WHIM, VHSE, etc.) in the protr package.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `extrPCMDescScales` for generalized AA property based scales descriptors, and `extrPCMPropScales` for (19 classes) AA descriptor based scales descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
data(AAindex)
AAidxmat = t(na.omit(as.matrix(AAindex[, 7:26])))
scales = extrPCMScales(x, propmat = AAidxmat, pc = 5, lag = 7, silent = FALSE)
```

---

extrProtAAC *Amino Acid Composition Descriptor*

Description

Amino Acid Composition Descriptor

Usage

```
extrProtAAC(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Amino Acid Composition descriptor (Dim: 20).

Value

A length 20 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See `extrProtDC` and `extrProtTC` for Dipeptide Composition and Tripeptide Composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtAAC(x)
```

`extrProtAPAAC` *Amphiphilic Pseudo Amino Acid Composition Descriptor*

Description

Amphiphilic Pseudo Amino Acid Composition Descriptor

Usage

```
extrProtAPAAC(x, props = c("Hydrophobicity", "Hydrophilicity"), lambda = 30,
  w = 0.05, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the properties used. 2 properties are used by default, as listed below: `'Hydrophobicity'` Hydrophobicity value of the 20 amino acids `'Hydrophilicity'` Hydrophilicity value of the 20 amino acids
`lambda`	The lambda parameter for the APAAC descriptors, default is 30.
`w`	The weighting factor, default is 0.05.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Amphiphilic Pseudo Amino Acid Composition (APAAC) descriptor (Dim: `20 + (n * lambda)`, `n` is the number of properties selected, default is 80).

Value

A length `20 + n * lambda` named vector, `n` is the number of properties selected.

Note

Note the default `20 * 2` `prop` values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the `AAindex` data, with or without the default three properties, which means users should explicitly specify the properties to use.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. *PROTEINS: Structure, Function, and Genetics*, 2001, 43: 246-255.

Type 2 pseudo amino acid composition. http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type2.htm

Kuo-Chen Chou. Using Amphiphilic Pseudo Amino Acid Composition to Predict Enzyme Subfamily Classes. *Bioinformatics*, 2005, 21, 10-19.

JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)

PNAS, 1981, 78:3824-3828 (T.P.Hopp & K.R.Woods). (The hydrophilicity data)

See Also

See `extrProtPAAC` for pseudo amino acid composition descriptor.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtAPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 2 default properties, 4 properties in the AAindex database,
# and 3 cutomized properties
extrProtAPAAC(x, customprops = myprops,
                 props = c('Hydrophobicity', 'Hydrophilicity',
                           'CIDH920105', 'BHAR880101',
                           'CHAM820101', 'CHAM820102',
                           'MyProp1', 'MyProp2', 'MyProp3'))
```

extrProtCTDC *CTD Descriptors - Composition*

Description

CTD Descriptors - Composition

Usage

```
extrProtCTDC(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Composition descriptor of the CTD descriptors (Dim: 21).

Value

A length 21 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See extrProtCTDT and extrProtCTDD for Transition and Distribution of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtCTDC(x)
```

`extrProtCTDCClass` *CTD Descriptors - Composition (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Composition (with Customized Amino Acid Classification Support)

Usage

```
extrProtCTDCClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Composition descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 3` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See extrProtCTDTClass and extrProtCTDDClass for Transition and Distribution of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extrProtCTDCClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

---

extrProtCTDD *CTD Descriptors - Distribution*

Description

CTD Descriptors - Distribution

Usage

```
extrProtCTDD(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Distribution descriptor of the CTD descriptors (Dim: 105).

Value

A length 105 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extrProtCTDC` and `extrProtCTDT` for Composition and Transition of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtCTDD(x)
```

`extrProtCTDDClass` *CTD Descriptors - Distribution (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Distribution (with Customized Amino Acid Classification Support)

Usage

```
extrProtCTDDClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Distribution descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 15` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extrProtCTDCClass` and `extrProtCTDTClass` for Composition and Transition of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extrProtCTDDClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

`extrProtCTDT` *CTD Descriptors - Transition*

Description

CTD Descriptors - Transition

Usage

```
extrProtCTDT(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function calculates the Transition descriptor of the CTD descriptors (Dim: 21).

Value

A length 21 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See `extrProtCTDC` and `extrProtCTDD` for Composition and Distribution of the CTD descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtCTDT(x)
```

`extrProtCTDTClass` *CTD Descriptors - Transition (with Customized Amino Acid Classification Support)*

Description

CTD Descriptors - Transition (with Customized Amino Acid Classification Support)

Usage

```
extrProtCTDTClass(x, aagroup1, aagroup2, aagroup3)
```

Arguments

`x`	A character vector, as the input protein sequence.
`aagroup1`	A named list which contains the first group of customized amino acid classification. See example below.
`aagroup2`	A named list which contains the second group of customized amino acid classification. See example below.
`aagroup3`	A named list which contains the third group of customized amino acid classification. See example below.

Details

This function calculates the Transition descriptor of the CTD descriptors, with customized amino acid classification support.

Value

A length `k * 3` named vector, `k` is the number of amino acid properties used.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

References

Inna Dubchak, Ilya Muchink, Stephen R. Holbrook and Sung-Hou Kim. Prediction of protein folding class using global description of amino acid sequence. *Proceedings of the National Academy of Sciences*. USA, 1995, 92, 8700-8704.

Inna Dubchak, Ilya Muchink, Christopher Mayor, Igor Dralyuk and Sung-Hou Kim. Recognition of a Protein Fold in the Context of the SCOP classification. *Proteins: Structure, Function and Genetics*, 1999, 35, 401-407.

See Also

See extrProtCTDCClass and extrProtCTDDClass for Composition and Distribution of the CTD descriptors with customized amino acid classification support.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]

# using five customized amino acid property classification
group1 = list(hydrophobicity  = c('R', 'K', 'E', 'D', 'Q', 'N'),
              normwaalsvolume = c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
              polarizability  = c('G', 'A', 'S', 'D', 'T'),
              secondarystruct = c('E', 'A', 'L', 'M', 'Q', 'K', 'R', 'H'),
              solventaccess   = c('A', 'L', 'F', 'C', 'G', 'I', 'V', 'W'))

group2 = list(hydrophobicity  = c('G', 'A', 'S', 'T', 'P', 'H', 'Y'),
              normwaalsvolume = c('N', 'V', 'E', 'Q', 'I', 'L'),
              polarizability  = c('C', 'P', 'N', 'V', 'E', 'Q', 'I', 'L'),
              secondarystruct = c('V', 'I', 'Y', 'C', 'W', 'F', 'T'),
              solventaccess   = c('R', 'K', 'Q', 'E', 'N', 'D'))

group3 = list(hydrophobicity  = c('C', 'L', 'V', 'I', 'M', 'F', 'W'),
              normwaalsvolume = c('M', 'H', 'K', 'F', 'R', 'Y', 'W'),
              polarizability  = c('K', 'M', 'H', 'F', 'R', 'Y', 'W'),
              secondarystruct = c('G', 'N', 'P', 'S', 'D'),
              solventaccess   = c('M', 'S', 'P', 'T', 'H', 'Y'))

extrProtCTDTClass(x, aagroup1 = group1, aagroup2 = group2, aagroup3 = group3)
```

---

extrProtCTriad *Conjoint Triad Descriptor*

Description

Conjoint Triad Descriptor

Usage

```
extrProtCTriad(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Conjoint Triad descriptor (Dim: 343).

Value

A length 343 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

J.W. Shen, J. Zhang, X.M. Luo, W.L. Zhu, K.Q. Yu, K.X. Chen, Y.X. Li, H.L. Jiang. Predicting Protein-protein Interactions Based Only on Sequences Information. *Proceedings of the National Academy of Sciences*. 007, 104, 4337–4341.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtCTriad(x)
```

---

extrProtCTriadClass *Conjoint Triad Descriptor (with Customized Amino Acid Classification Support)*

Description

Conjoint Triad Descriptor (with Customized Amino Acid Classification Support)

Usage

```
extrProtCTriadClass(x, aaclass)
```

Arguments

x	A character vector, as the input protein sequence.
aaclass	A list containing the customized amino acid classification. See example below.

Details

This function calculates the Conjoint Triad descriptor, with customized amino acid classification support.

Value

A length k^3 named vector, where k is the number of customized classes of the amino acids.

Note

For this descriptor type, users need to intelligently evaluate the underlying details of the descriptors provided, instead of using this function with their data blindly. It would be wise to use some negative and positive control comparisons where relevant to help guide interpretation of the results.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

References

J.W. Shen, J. Zhang, X.M. Luo, W.L. Zhu, K.Q. Yu, K.X. Chen, Y.X. Li, H.L. Jiang. Predicting Protein-protein Interactions Based Only on Sequences Information. *Proceedings of the National Academy of Sciences*. 007, 104, 4337–4341.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]

# using customized amino acid classification (normalized van der Waals volume)
newclass = list(c('G', 'A', 'S', 'T', 'P', 'D', 'C'),
                c('N', 'V', 'E', 'Q', 'I', 'L'),
                c('M', 'H', 'K', 'F', 'R', 'Y', 'W'))

extrProtCTriadClass(x, aaclass = newclass)
```

---

extrProtDC *Dipeptide Composition Descriptor*

Description

Dipeptide Composition Descriptor

Usage

```
extrProtDC(x)
```

Arguments

x	A character vector, as the input protein sequence.

Details

This function calculates the Dipeptide Composition descriptor (Dim: 400).

Value

A length 400 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See extrProtTC for tripeptide composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtDC(x)
```

`extrProtFPGap` *Amino Acid Properties Based Scales Descriptors (Protein Fingerprint) with Gap Support*

Description

Amino Acid Properties Based Scales Descriptors (Protein Fingerprint) with Gap Support

Usage

```
extrProtFPGap(x, index = NULL, pc, lag, scale = TRUE, silent = TRUE)
```

Arguments

`x`	A character vector, as the input protein sequence. Use '-' to represent gaps in the sequence.
`index`	Integer vector or character vector. Specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database. Default is `NULL`, means selecting all the AA properties in the AAindex database.
`pc`	Integer. Use the first pc principal components as the scales. Must be no greater than the number of AA properties provided.
`lag`	The lag parameter. Must be less than the amino acids.
`scale`	Logical. Should we auto-scale the property matrix before PCA? Default is `TRUE`.
`silent`	Logical. Whether we print the standard deviation, proportion of variance and the cumulative proportion of the selected principal components or not. Default is `TRUE`.

Details

This function calculates amino acid properties based scales descriptors (protein fingerprint) with gap support. Users could specify which AAindex properties to select from the AAindex database by specify the numerical or character index of the properties in the AAindex database.

Value

A length `lag * p^2` named vector, p is the number of scales (principal components) selected.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
# amino acid sequence with gaps
x = readFASTA(system.file('protseq/align.fasta', package = 'BioMedR'))$`IXI_235`
fp = extrProtFPGap(x, index = c(160:165, 258:296), pc = 5, lag = 7, silent = FALSE)
```

`extrProtGeary` *Geary Autocorrelation Descriptor*

Description

Geary Autocorrelation Descriptor

Usage

```
extrProtGeary(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below: **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992) **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988) **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982) **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982) **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976) **AccNo. BIGC670101** Residue volume (Bigelow, 1967) **AccNo. CHAM810101** Steric parameter (Charton, 1981) **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
`nlag`	Maximum value of the lag parameter. Default is `30`.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Geary autocorrelation descriptor (Dim: `length(props) * nlag`).

Value

A length `nlag` named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See `extrProtMoreauBroto` and `extrProtMoran` for Moreau-Broto autocorrelation descriptors and Moran autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtGeary(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 4 properties in the AAindex database, and 3 cutomized properties
extrProtGeary(x, customprops = myprops,
                 props = c('CIDH920105', 'BHAR880101',
                           'CHAM820101', 'CHAM820102',
                           'MyProp1', 'MyProp2', 'MyProp3'))
```

`extrProtMoran` *Moran Autocorrelation Descriptor*

Description

Moran Autocorrelation Descriptor

Usage

```
extrProtMoran(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

x	A character vector, as the input protein sequence.
props	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below: **AccNo. CIDH920105** Normalized average hydrophobicity scales (Cid et al., 1992) **AccNo. BHAR880101** Average flexibility indices (Bhaskaran-Ponnuswamy, 1988) **AccNo. CHAM820101** Polarizability parameter (Charton-Charton, 1982) **AccNo. CHAM820102** Free energy of solution in water, kcal/mole (Charton-Charton, 1982) **AccNo. CHOC760101** Residue accessible surface area in tripeptide (Chothia, 1976) **AccNo. BIGC670101** Residue volume (Bigelow, 1967) **AccNo. CHAM810101** Steric parameter (Charton, 1981) **AccNo. DAYM780201** Relative mutability (Dayhoff et al., 1978b)
nlag	Maximum value of the lag parameter. Default is `30`.
customprops	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Moran autocorrelation descriptor (Dim: `length(props) * nlag`).

Value

A length `nlag` named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See `extrProtMoreauBroto` and `extrProtGeary` for Moreau-Broto autocorrelation descriptors and Geary autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtMoran(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 4 properties in the AAindex database, and 3 cutomized properties
extrProtMoran(x, customprops = myprops,
                 props = c('CIDH920105', 'BHAR880101',
                           'CHAM820101', 'CHAM820102',
                           'MyProp1', 'MyProp2', 'MyProp3'))
```

---

extrProtMoreauBroto *Normalized Moreau-Broto Autocorrelation Descriptor*

Description

Normalized Moreau-Broto Autocorrelation Descriptor

Usage

```
extrProtMoreauBroto(x, props = c("CIDH920105", "BHAR880101", "CHAM820101",
  "CHAM820102", "CHOC760101", "BIGC670101", "CHAM810101", "DAYM780201"),
  nlag = 30L, customprops = NULL)
```

Arguments

x	A character vector, as the input protein sequence.
props	A character vector, specifying the Accession Number of the target properties. 8 properties are used by default, as listed below:

AccNo. CIDH920105 Normalized average hydrophobicity scales (Cid et al., 1992)

AccNo. BHAR880101 Average flexibility indices (Bhaskaran-Ponnuswamy, 1988)

AccNo. CHAM820101 Polarizability parameter (Charton-Charton, 1982)

AccNo. CHAM820102 Free energy of solution in water, kcal/mole (Charton-Charton, 1982)

AccNo. CHOC760101 Residue accessible surface area in tripeptide (Chothia, 1976)

AccNo. BIGC670101 Residue volume (Bigelow, 1967)

AccNo. CHAM810101 Steric parameter (Charton, 1981)

AccNo. DAYM780201 Relative mutability (Dayhoff et al., 1978b)

nlag	Maximum value of the lag parameter. Default is 30.
customprops	A n x 21 named data frame contains n customize property. Each row contains one property. The column order for different amino acid types is 'AccNo', 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V', and the columns should also be *exactly* named like this. The AccNo column contains the properties' names. Then users should explicitly specify these properties with these names in the argument props. See the examples below for a demonstration. The default value for customprops is NULL.

Details

This function calculates the normalized Moreau-Broto autocorrelation descriptor (Dim: length(props) * nlag).

Value

A length nlag named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

AAindex: Amino acid index database. http://www.genome.ad.jp/dbget/aaindex.html

Feng, Z.P. and Zhang, C.T. (2000) Prediction of membrane protein types based on the hydrophobic index of amino acids. *Journal of Protein Chemistry*, 19, 269-275.

Horne, D.S. (1988) Prediction of protein helix content from an autocorrelation analysis of sequence hydrophobicities. *Biopolymers*, 27, 451-477.

Sokal, R.R. and Thomson, B.A. (2006) Population structure inferred by local spatial autocorrelation: an Usage from an Amerindian tribal population. *American Journal of Physical Anthropology*, 129, 121-131.

See Also

See extrProtMoran and extrProtGeary for Moran autocorrelation descriptors and Geary autocorrelation descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtMoreauBroto(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))
```

```
# Use 4 properties in the AAindex database, and 3 cutomized properties
extrProtMoreauBroto(x, customprops = myprops,
                       props = c('CIDH920105', 'BHAR880101',
                                 'CHAM820101', 'CHAM820102',
                                 'MyProp1', 'MyProp2', 'MyProp3'))
```

`extrProtPAAC` *Pseudo Amino Acid Composition Descriptor*

Description

Pseudo Amino Acid Composition Descriptor

Usage

```
extrProtPAAC(x, props = c("Hydrophobicity", "Hydrophilicity",
  "SideChainMass"), lambda = 30, w = 0.05, customprops = NULL)
```

Arguments

`x`	A character vector, as the input protein sequence.
`props`	A character vector, specifying the properties used. 3 properties are used by default, as listed below: `'Hydrophobicity'` Hydrophobicity value of the 20 amino acids `'Hydrophilicity'` Hydrophilicity value of the 20 amino acids `'SideChainMass'` Side-chain mass of the 20 amino acids
`lambda`	The lambda parameter for the PAAC descriptors, default is 30.
`w`	The weighting factor, default is 0.05.
`customprops`	A `n x 21` named data frame contains `n` customize property. Each row contains one property. The column order for different amino acid types is `'AccNo'`, `'A'`, `'R'`, `'N'`, `'D'`, `'C'`, `'E'`, `'Q'`, `'G'`, `'H'`, `'I'`, `'L'`, `'K'`, `'M'`, `'F'`, `'P'`, `'S'`, `'T'`, `'W'`, `'Y'`, `'V'`, and the columns should also be *exactly* named like this. The `AccNo` column contains the properties' names. Then users should explicitly specify these properties with these names in the argument `props`. See the examples below for a demonstration. The default value for `customprops` is `NULL`.

Details

This function calculates the Pseudo Amino Acid Composition (PAAC) descriptor (Dim: `20 + lambda`, default is 50).

Value

A length `20 + lambda` named vector

Note

Note the default `20 * 3` `prop` values have been already independently given in the function. Users could also specify other (up to 544) properties with the Accession Number in the `AAindex` data, with or without the default three properties, which means users should explicitly specify the properties to use.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition. *PROTEINS: Structure, Function, and Genetics*, 2001, 43: 246-255.

Type 1 pseudo amino acid composition. http://www.csbio.sjtu.edu.cn/bioinf/PseAAC/type1.htm

Kuo-Chen Chou. Using Amphiphilic Pseudo Amino Acid Composition to Predict Enzyme Subfamily Classes. *Bioinformatics*, 2005, 21, 10-19.

JACS, 1962, 84: 4240-4246. (C. Tanford). (The hydrophobicity data)

PNAS, 1981, 78:3824-3828 (T.P.Hopp & K.R.Woods). (The hydrophilicity data)

CRC Handbook of Chemistry and Physics, 66th ed., CRC Press, Boca Raton, Florida (1985). (The side-chain mass data)

R.M.C. Dawson, D.C. Elliott, W.H. Elliott, K.M. Jones, Data for Biochemical Research 3rd ed., Clarendon Press Oxford (1986). (The side-chain mass data)

See Also

See extrProtAPAAC for amphiphilic pseudo amino acid composition descriptor.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtPAAC(x)

myprops = data.frame(AccNo = c("MyProp1", "MyProp2", "MyProp3"),
                     A = c(0.62,  -0.5, 15),  R = c(-2.53,   3, 101),
                     N = c(-0.78,  0.2, 58),  D = c(-0.9,    3, 59),
                     C = c(0.29,    -1, 47),  E = c(-0.74,   3, 73),
                     Q = c(-0.85,  0.2, 72),  G = c(0.48,    0, 1),
                     H = c(-0.4,  -0.5, 82),  I = c(1.38, -1.8, 57),
                     L = c(1.06,  -1.8, 57),  K = c(-1.5,    3, 73),
                     M = c(0.64,  -1.3, 75),  F = c(1.19, -2.5, 91),
                     P = c(0.12,     0, 42),  S = c(-0.18, 0.3, 31),
                     T = c(-0.05, -0.4, 45),  W = c(0.81, -3.4, 130),
                     Y = c(0.26,  -2.3, 107), V = c(1.08, -1.5, 43))

# Use 3 default properties, 4 properties in the AAindex database,
# and 3 cutomized properties
extrProtPAAC(x, customprops = myprops,
               props = c('Hydrophobicity', 'Hydrophilicity', 'SideChainMass',
                         'CIDH920105', 'BHAR880101',
                         'CHAM820101', 'CHAM820102',
                         'MyProp1', 'MyProp2', 'MyProp3'))
```

`extrProtPSSM` *Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence*

Description

Compute PSSM (Position-Specific Scoring Matrix) for given protein sequence

Usage

```
extrProtPSSM(seq, start.pos = 1L, end.pos = nchar(seq),
  psiblast.path = NULL, makeblastdb.path = NULL, database.path = NULL,
  iter = 5, silent = TRUE, evalue = 10L, word.size = NULL,
  gapopen = NULL, gapextend = NULL, matrix = "BLOSUM62",
  threshold = NULL, seg = "no", soft.masking = FALSE,
  culling.limit = NULL, best.hit.overhang = NULL,
  best.hit.score.edge = NULL, xdrop.ungap = NULL, xdrop.gap = NULL,
  xdrop.gap.final = NULL, window.size = NULL, gap.trigger = 22L,
  num.threads = 1L, pseudocount = 0L, inclusion.ethresh = 0.002)
```

Arguments

`seq`	Character vector, as the input protein sequence.
`start.pos`	Optional integer denoting the start position of the fragment window. Default is 1, i.e. the first amino acid of the given sequence.
`end.pos`	Optional integer denoting the end position of the fragment window. Default is `nchar(seq)`, i.e. the last amino acid of the given sequence.
`psiblast.path`	Character string indicating the path of the `psiblast` program. If NCBI Blast+ was previously installed in the operation system, the path will be automatically detected.
`makeblastdb.path`	Character string indicating the path of the `makeblastdb` program. If NCBI Blast+ was previously installed in the system, the path will be automatically detected.
`database.path`	Character string indicating the path of a reference database (a FASTA file).
`iter`	Number of iterations to perform for PSI-Blast.
`silent`	Logical. Whether the PSI-Blast running output should be shown or not (May not work on some Windows versions and PSI-Blast versions), default is `TRUE`.
`evalue`	Expectation value (E) threshold for saving hits. Default is `10`.
`word.size`	Word size for wordfinder algorithm. An integer >= 2.
`gapopen`	Integer. Cost to open a gap.
`gapextend`	Integer. Cost to extend a gap.
`matrix`	Character string. The scoring matrix name (default is `'BLOSUM62'`).
`threshold`	Minimum word score such that the word is added to the BLAST lookup table. A real value >= 0.
`seg`	Character string. Filter query sequence with SEG (`'yes'`, `'window locut hicut'`, or `'no'` to disable) Default is `'no'`.

`soft.masking` Logical. Apply filtering locations as soft masks? Default is `FALSE`.

`culling.limit` An integer >= 0. If the query range of a hit is enveloped by that of at least this many higher-scoring hits, delete the hit. Incompatible with `best.hit.overhang` and `best_hit_score_edge`.

`best.hit.overhang`
Best Hit algorithm overhang value (A real value >= 0 and =< 0.5, recommended value: 0.1). Incompatible with `culling_limit`.

`best.hit.score.edge`
Best Hit algorithm score edge value (A real value >=0 and =< 0.5, recommended value: 0.1). Incompatible with `culling_limit`.

`xdrop.ungap` X-dropoff value (in bits) for ungapped extensions.

`xdrop.gap` X-dropoff value (in bits) for preliminary gapped extensions.

`xdrop.gap.final`
X-dropoff value (in bits) for final gapped alignment.

`window.size` An integer >= 0. Multiple hits window size, To specify 1-hit algorithm, use `0`.

`gap.trigger` Number of bits to trigger gapping. Default is `22`.

`num.threads` Integer. Number of threads (CPUs) to use in the BLAST search. Default is `1`.

`pseudocount` Integer. Pseudo-count value used when constructing PSSM. Default is `0`.

`inclusion.ethresh`
E-value inclusion threshold for pairwise alignments. Default is `0.002`.

Details

This function calculates the PSSM (Position-Specific Scoring Matrix) derived by PSI-Blast for given protein sequence or peptides. For given protein sequences or peptides, PSSM represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Note that the output value is not normalized.

Value

The original PSSM, a numeric matrix which has `end.pos - start.pos + 1` columns and `20` named rows.

Note

The function requires the `makeblastdb` and `psiblast` programs to be properly installed in the operation system or their paths provided.

The two command-line programs are included in the NCBI-BLAST+ software package. To install NCBI Blast+, just open the NCBI FTP site using web browser or FTP software: `ftp://anonymous@ftp.ncbi.nlm.nih.gov:21/blast/executables/blast+/LATEST/` then download the executable version of BLAST+ according to your operation system, and compile or install the downloaded source code or executable program.

Ubuntu/Debian users can directly use the command `sudo apt-get install ncbi-blast+` to install NCBI Blast+. For OS X users, download `ncbi-blast- ... .dmg` then install. For Windows users, download `ncbi-blast- ... .exe` then install.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Altschul, Stephen F., et al. "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs." *Nucleic acids research* 25.17 (1997): 3389–3402.

Ye, Xugang, Guoli Wang, and Stephen F. Altschul. "An assessment of substitution scores for protein profile-profile comparison." *Bioinformatics* 27.24 (2011): 3356–3363.

Rangwala, Huzefa, and George Karypis. "Profile-based direct kernels for remote homology detection and fold recognition." *Bioinformatics* 21.23 (2005): 4239–4247.

See Also

extrProtPSSMFeature extrProtPSSMAcc

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'BioMedR'), to = dbpath))
pssmmat = extrProtPSSM(seq = x, database.path = dbpath)
dim(pssmmat)  # 20 x 562 (P00750: length 562, 20 Amino Acids)
```

---

extrProtPSSMAcc *Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix) and auto cross covariance*

Description

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix) and auto cross covariance

Usage

```
extrProtPSSMAcc(pssmmat, lag)
```

Arguments

pssmmat	The PSSM computed by extrProtPSSM.
lag	The lag parameter. Must be less than the number of amino acids in the sequence (i.e. the number of columns in the PSSM matrix).

Details

This function calculates the feature vector based on the PSSM by running PSI-Blast and auto cross covariance tranformation.

Value

A length `lag * 20^2` named numeric vector, the element names are derived by the amino acid name abbreviation (crossed amino acid name abbreviation) and lag index.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Wold, S., Jonsson, J., Sj\"orstr\"om, M., Sandberg, M., & R\"annar, S. (1993). DNA and peptide sequences and chemical processes multivariately modelled by principal component analysis and partial least-squares projections to latent structures. *Analytica chimica acta*, 277(2), 239–253.

See Also

extrProtPSSM extrProtPSSMFeature

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'BioMedR'), to = dbpath))
pssmmat = extrProtPSSM(seq = x, database.path = dbpath)
pssmacc = extrProtPSSMAcc(pssmmat, lag = 3)
tail(pssmacc)
```

---

extrProtPSSMFeature *Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)*

Description

Profile-based protein representation derived by PSSM (Position-Specific Scoring Matrix)

Usage

```
extrProtPSSMFeature(pssmmat)
```

Arguments

pssmmat	The PSSM computed by extrProtPSSM.

Details

This function calculates the profile-based protein representation derived by PSSM. The feature vector is based on the PSSM computed by extrProtPSSM. For a given sequence, The PSSM feature represents the log-likelihood of the substitution of the 20 types of amino acids at that position in the sequence. Each PSSM feature value in the vector represents the degree of conservation of a given amino acid type. The value is normalized to interval (0, 1) by the transformation 1/(1+e^(-x)).

Value

A numeric vector which has 20 x N named elements, where N is the size of the window (number of rows of the PSSM).

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Ye, Xugang, Guoli Wang, and Stephen F. Altschul. "An assessment of substitution scores for protein profile-profile comparison." *Bioinformatics* 27.24 (2011): 3356–3363.

Rangwala, Huzefa, and George Karypis. "Profile-based direct kernels for remote homology detection and fold recognition." *Bioinformatics* 21.23 (2005): 4239–4247.

See Also

extrProtPSSM extrProtPSSMAcc

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
dbpath = tempfile('tempdb', fileext = '.fasta')
invisible(file.copy(from = system.file('protseq/Plasminogen.fasta', package = 'BioMedR'), to = dbpath))
pssmmat = extractProtPSSM(seq = x, database.path = dbpath)
pssmfeature = extrProtPSSMFeature(pssmmat)
head(pssmfeature)
```

---

extrProtQSO *Quasi-Sequence-Order (QSO) Descriptor*

Description

Quasi-Sequence-Order (QSO) Descriptor

Usage

```
extrProtQSO(x, nlag = 30, w = 0.1)
```

Arguments

x	A character vector, as the input protein sequence.
nlag	The maximum lag, defualt is 30.
w	The weighting factor, default is 0.1.

Details

This function calculates the Quasi-Sequence-Order (QSO) descriptor (Dim: `20 + 20 + (2 * nlag)`, default is 100).

Value

A length `20 + 20 + (2 * nlag)` named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. *Biochemical and Biophysical Research Communications*, 2000, 278, 477-483.

Kuo-Chen Chou and Yu-Dong Cai. Prediction of Protein Sucellular Locations by GO-FunD-PseAA Predictor. *Biochemical and Biophysical Research Communications*, 2004, 320, 1236-1239.

Gisbert Schneider and Paul Wrede. The Rational Design of Amino Acid Sequences by Artifical Neural Networks and Simulated Molecular Evolution: Do Novo Design of an Idealized Leader Cleavge Site. *Biophys Journal*, 1994, 66, 335-344.

See Also

See `extrProtSOCN` for sequence-order-coupling numbers.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtQSO(x)
```

`extrProtSOCN` *Sequence-Order-Coupling Numbers*

Description

Sequence-Order-Coupling Numbers

Usage

```
extrProtSOCN(x, nlag = 30)
```

Arguments

`x`	A character vector, as the input protein sequence.
`nlag`	The maximum lag, defualt is 30.

Details

This function calculates the Sequence-Order-Coupling Numbers (Dim: `nlag * 2`, default is 60).

Value

A length `nlag * 2` named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Kuo-Chen Chou. Prediction of Protein Subcellar Locations by Incorporating Quasi-Sequence-Order Effect. *Biochemical and Biophysical Research Communications*, 2000, 278, 477-483.

Kuo-Chen Chou and Yu-Dong Cai. Prediction of Protein Sucellular Locations by GO-FunD-PseAA Predictor. *Biochemical and Biophysical Research Communications*, 2004, 320, 1236-1239.

Gisbert Schneider and Paul Wrede. The Rational Design of Amino Acid Sequences by Artifical Neural Networks and Simulated Molecular Evolution: Do Novo Design of an Idealized Leader Cleavge Site. *Biophys Journal*, 1994, 66, 335-344.

See Also

See `extrProtQSO` for quasi-sequence-order descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtSOCN(x)
```

`extrProtTC` *Tripeptide Composition Descriptor*

Description

Tripeptide Composition Descriptor

Usage

```
extrProtTC(x)
```

Arguments

`x`	A character vector, as the input protein sequence.

Details

This function calculates the Tripeptide Composition descriptor (Dim: 8000).

Value

A length 8000 named vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

M. Bhasin, G. P. S. Raghava. Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition. *Journal of Biological Chemistry*, 2004, 279, 23262.

See Also

See extrProtDC for dipeptide composition descriptors.

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
extrProtTC(x)
```

---

geometric *Descriptor Characterizing the Mass Distribution of the Molecule.*

Description

Descriptor Characterizing the Mass Distribution of the Molecule.

Calculates the Ratio of Length to Breadth Descriptor

Descriptor that Calculates the Principal Moments of Inertia and Ratios of the Principal Moments

Usage

```
extrDrugGravitationalIndex(molecules, silent = TRUE)

extrDrugLengthOverBreadth(molecules, silent = TRUE)

extrDrugMomentOfInertia(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Descriptor characterizing the mass distribution of the molecule described by Katritzky et al. For modelling purposes the value of the descriptor is calculated both with and without H atoms. Furthermore the square and cube roots of the descriptor are also generated as described by Wessel et al.

Calculates the Ratio of Length to Breadth, as a result ti does not perform any orientation and only considers the X & Y extents for a series of rotations about the Z axis (in 10 degree increments).

A descriptor that calculates the moment of inertia and radius of gyration. Moment of inertia (MI) values characterize the mass distribution of a molecule. Related to the MI values, ratios of the MI values along the three principal axes are also well know modeling variables. This descriptor calculates the MI values along the X, Y and Z axes as well as the ratio's X/Y, X/Z and Y/Z. Finally it also calculates the radius of gyration of the molecule.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 9 columns:

- GRAV.1 - gravitational index of heavy atoms
- GRAV.2 - square root of gravitational index of heavy atoms
- GRAV.3 - cube root of gravitational index of heavy atoms
- GRAVH.1 - gravitational index - hydrogens included
- GRAVH.2 - square root of hydrogen-included gravitational index
- GRAVH.3 - cube root of hydrogen-included gravitational index
- GRAV.4 - grav1 for all pairs of atoms (not just bonded pairs)
- GRAV.5 - grav2 for all pairs of atoms (not just bonded pairs)
- GRAV.6 - grav3 for all pairs of atoms (not just bonded pairs)

extrDrugLengthOverBreadth: This function returns two columns named LOBMAX and LOBMIN:

- LOBMAX - The maximum L/B ratio;
- LOBMIN - The L/B ratio for the rotation that results in the minimum area (defined by the product of the X & Y extents for that orientation).

extrDrugMomentOfInertia: This function returns 7 columns named MOMI.X, MOMI.Y, MOMI.Z, MOMI.XY, MOMI.XZ, MOMI.YZ, MOMI.R:

- MOMI.X - MI along X axis
- MOMI.Y - MI along Y axis
- MOMI.Z - MI along Z axis
- MOMI.XY - X/Y
- MOMI.XZ - X/Z
- MOMI.YZ - Y/Z
- MOMI.R - Radius of gyration

One important aspect of the algorithm is that if the eigenvalues of the MI tensor are below 1e-3, then the ratio's are set to a default of 1000.

Note

extrDrugLengthOverBreadth : The descriptor assumes that the atoms have been configured.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Katritzky, A.R. and Mu, L. and Lobanov, V.S. and Karelson, M., Correlation of Boiling Points With Molecular Structure. 1. A Training Set of 298 Diverse Organics and a Test Set of 9 Simple Inorganics, J. Phys. Chem., 1996, 100:10400-10407.

Wessel, M.D. and Jurs, P.C. and Tolan, J.W. and Muskal, S.M. , Prediction of Human Intestinal Absorption of Drug Compounds From Molecular Structure, Journal of Chemical Information and Computer Sciences, 1998, 38:726-735.

Examples

```
sdf = system.file('sysdata/test.sdf', package = 'BioMedR')
mol = readMolFromSDF(sdf)
# Descriptor Characterizing the Mass Distribution of the Molecule
dat = extrDrugGravitationalIndex(mol)
head(dat)
# Calculates the Ratio of Length to Breadth Descriptor
dat = extrDrugLengthOverBreadth(mol)
head(dat)
# Descriptor that Calculates the Principal Moments of
# Inertia and Ratios of the Principal Moments
dat = extrDrugMomentOfInertia(mol)
head(dat)
```

---

getCPI *Generating Interaction Descriptors*

Description

Generating Interaction Descriptors

Usage

```
getCPI(drugmat, protmat, type = c("combine", "tensorprod"))

getPPI(protmat1, protmat2, type = c("combine", "tensorprod", "entrywise"))

getDDI(DNAmat1, DNAmat2, type = c("combine", "tensorprod", "entrywise"))

getDPI(DNAmat, protmat, type = c("combine", "tensorprod"))

getCCI(drugmat1, drugmat2, type = c("combine", "tensorprod", "entrywise"))

getCDI(drugmat, DNAmat, type = c("combine", "tensorprod"))
```

Arguments

drugmat	The compound descriptor matrix.
protmat	The protein descriptor matrix.
type	The interaction type, one or more of "combine", "tensorprod", and "entrywise".
protmat1	The first protein descriptor matrix, must have the same ncol with protmat2.
protmat2	The second protein descriptor matrix, must have the same ncol with protmat1.
DNAmat1	The first DNA descriptor matrix, must have the same ncol with DNAmat2.
DNAmat2	The second DNA descriptor matrix, must have the same ncol with DNAmat1.
DNAmat	The DNA descriptor matrix.
drugmat1	The first compound descriptor matrix, must have the same ncol with drugmat2.
drugmat2	The second compound descriptor matrix, must have the same ncol with drugmat1.

Details

This function calculates the interaction descriptors by three types of interaction:

- `combine` - combine the two descriptor matrix, result has `(p1 + p2)` columns
- `tensorprod` - calculate column-by-column (pseudo)-tensor product type interactions, result has `(p1 * p2)` columns
- `entrywise` - calculate entrywise product and entrywise sum of the two matrices, then combine them, result has `(p + p)` columns

Value

A matrix containing the interaction descriptors

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
x = matrix(1:10, ncol = 2)
y = matrix(1:15, ncol = 3)
# getCPI
getCPI(x, y, 'combine')
# getCDI
getCDI(x, y, 'tensorprod')
# getDPI
getDPI(x, y, type = c('combine', 'tensorprod'))
getDPI(x, y, type = c('tensorprod', 'combine'))
x = matrix(1:10, ncol = 2)
y = matrix(5:14, ncol = 2)

# getPPI
getPPI(x, y, type = 'combine')
getPPI(x, y, type = 'tensorprod')
# getDDI
getDDI(x, y, type = 'entrywise')
getDDI(x, y, type = c('combine', 'tensorprod'))
# getCCI
getCCI(x, y, type = c('combine', 'entrywise'))
getCCI(x, y, type = c('entrywise', 'tensorprod'))
getCCI(x, y, type = c('combine', 'entrywise', 'tensorprod'))
```

`getDrug` *Retrieve Drug Molecules in MOL and SMILES Format from Databases*

Description

Retrieve Drug Molecules in MOL and SMILES Format from Databases(BMgetDrug)

Retrieve Drug Molecules in MOL and Smi Format from the PubChem Database(BMgetDrug...PubChem)

Retrieve Drug Molecules in MOL and Smi Format from the ChEMBL Database(BMgetDrug...ChEMBL)

Retrieve Drug Molecules in InChI Format from the CAS Database(BMDrugMolCAS)

Retrieve Drug Molecules in MOL and Smi Format from the KEGG Database(BMgetDrug...KEGG)

Retrieve Drug Molecules in MOL and Smi Format from the DrugBank Database(BMgetDrug...DrugBank)

Usage

```
BMgetDrug(id, from = c("pubchem", "chembl", "cas", "kegg", "drugbank"),
  type = c("mol", "smile"), parallel = 5)

BMgetDrugMolPubChem(id, parallel = 5)

BMgetDrugSmiPubChem(id, parallel = 5)

BMgetDrugMolChEMBL(id, parallel = 5)

BMgetDrugSmiChEMBL(id, parallel = 5)

BMDrugMolCAS(id, parallel = 5)

BMgetDrugMolKEGG(id, parallel = 5)

BMgetDrugSmiKEGG(id, parallel = 5)

BMgetDrugMolDrugBank(id, parallel = 5)

BMgetDrugSmiDrugBank(id, parallel = 5)
```

Arguments

id	A character vector, as the drug ID(s).
from	The database, one of 'pubchem', 'chembl', 'cas', 'kegg', 'drugbank'.
type	The returned molecule format, mol or smile.
parallel	An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves drug molecules in MOL and SMILES format from five databases.

This function retrieves drug molecules in MOL format from the PubChem database.

This function retrieves drug molecules in MOL format from the ChEMBL database.

This function retrieves drug molecules in InChI format from the CAS database. CAS database only provides InChI data, so here we return the molecule in InChI format, users could convert them to SMILES format using Open Babel (http://openbabel.org/) or other third-party tools.

This function retrieves drug molecules in MOL format from the KEGG database.

This function retrieves drug molecules in MOL format from the DrugBank database.

Value

A length of id character vector, each element containing the corresponding drug molecule.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See BMgetProt for retrieving protein sequences from three databases.

Examples

```
# BMgetDrug
id = c('DB00859', 'DB00860')
BMgetDrug(id, 'drugbank', 'smile')

# BMgetDrugMolPubChem
id = c('7847562', '7847563')  # Penicillamine
BMgetDrugMolPubChem(id)

# BMgetDrugsmiPubChem
id = c('7847562', '7847563')  # Penicillamine
BMgetDrugSmiPubChem(id)

# BMgetDrugMolChEMBL
id = 'CHEMBL1430'  # Penicillamine
BMgetDrugMolChEMBL(id)

# BMgetDrugSmiChEMBL
id = 'CHEMBL1430'  # Penicillamine
BMgetDrugSmiChEMBL(id)

# BMDrugMolCAS
id = '52-67-5'  # Penicillamine
BMDrugMolCAS(id)

# BMgetDrugMolKEGG
id = 'D00496'  # Penicillamine
BMgetDrugMolKEGG(id)

# BMgetDrugSmiKEGG
id = 'D00496'  # Penicillamine
BMgetDrugSmiKEGG(id)

# BMgetDrugMolDrugBank
id = 'DB00859'  # Penicillamine
BMgetDrugMolDrugBank(id)

# BMgetDrugSmiDrugBank
id = 'DB00859'  # Penicillamine
BMgetDrugSmiDrugBank(id)
```

---

getProt *Retrieve Protein Sequence in various Formats from Databases*

Description

Retrieve Protein Sequence in various Formats from Databases(BMgetDrug)

Retrieve Protein Sequence (FASTA Format) from the UniProt Database(BMgetProt...UinProt)

Retrieve Protein Sequence (FASTA Format) from the KEGG Database(BMgetProt...KEGG)

Retrieve Protein Sequence (PDB Format) from RCSB PDB(BMgetProt...RCSBPDB)

Usage

```
BMgetProt(id, from = c("uniprot", "kegg", "pdb"), type = c("fasta", "pdb",
  "aaseq"), parallel = 5)

BMgetProtFASTAUinProt(id, parallel = 5)

BMgetProtSeqUniProt(id, parallel = 5)

BMgetProtFASTAKEGG(id, parallel = 5)

BMgetProtSeqKEGG(id, parallel = 5)

BMgetProtPDBRCSBPDB(id, parallel = 5)

BMgetProtSeqRCSBPDB(id, parallel = 5)
```

Arguments

id	A character vector, as the protein ID(s).
from	The database, one of 'uniprot', 'kegg', 'pdb'.
type	The returned protein format, one of fasta, pdb, aaseq.
parallel	An integer, the parallel parameter, indicates how many process the user would like to use for retrieving the data (using RCurl), default is 5. For regular cases, we recommend a number less than 20.

Details

This function retrieves protein sequence in various formats from three databases.

This function retrieves protein sequences (FASTA format) from the UniProt database.

This function retrieves protein sequences (FASTA format) from the KEGG database.

This function retrieves protein sequences (PDB format) from RCSB PDB.

Value

A length of id character list, each element containing the corresponding protein sequence(s) or file(s).

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

UniProt. http://www.uniprot.org/

UniProt REST API Documentation. http://www.uniprot.org/faq/28

UniProt. http://www.uniprot.org/

UniProt REST API Documentation. http://www.uniprot.org/faq/28

See Also

See BMgetDrug for retrieving drug molecules from five databases.

Examples

```
# BMgetProt
id = c('P00750', 'P00751', 'P00752')
BMgetProt(id, from = 'uniprot', type = 'aaseq')

# BMgetProtFASTAUinProt
id = c('P00750', 'P00751', 'P00752')
BMgetProtFASTAUinProt(id)

# BMgetProtSeqUniProt
id = c('P00750', 'P00751', 'P00752')
BMgetProtSeqUniProt(id)

#  BMgetProtFASTAKEGG
id = c('hsa:10161', 'hsa:10162')
BMgetProtFASTAKEGG(id)

# BMgetProtSeqKEG
id = c('hsa:10161', 'hsa:10162')
BMgetProtSeqKEGG(id)

# BMgetProtPDBRCSBPDB
id = c('4HHB', '4FF9')
BMgetProtPDBRCSBPDB(id)

# BMgetProtSeqRCSBPDB
id = c('4HHB', '4FF9')
BMgetProtSeqRCSBPDB(id)
```

---

make_kmer_index *Calculate The Basic Kmer Feature Vector*

Description

Calculate The Basic Kmer Feature Vector

Usage

```
make_kmer_index(k, alphabet = "ACGT")
```

Arguments

k	the k value of kmer, it should be an integer larger than 0.
alphabet	the

Details

This function calculate the basic kmer feature vector.

Value

The result character vector

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See extrDNAkmer

Examples

```
make_kmer_index(2, alphabet = "ACGT")
```

NNeighbors *Nearest Neighbors*

Description

Nearest Neighbors

Usage

```
NNeighbors(x, numNbrs = NULL, cutoff = NULL, ...)
```

Arguments

x	Either an FPset or an APset.
numNbrs	Number of neighbors to find for each item. If not enough neighbors can be found the matrix will be padded with NA.
cutoff	The minimum similarity value an item must have to another item in order to be included in that items neighbor list. This parameter takes precedence over numNbrs. This parameter allows to obtain tighter clustering results.
...	These parameters will be passed into the distance function used

Details

Computes the nearest neighbors of descriptors in an FPset or APset object for use with the jarvisPatrick clustering function. Only one of numNbrs or cutoff should be given, cutoff will take precedence if both are given. If numNbrs is given, then that many neighbors will be returned for each item in the set. If cutoff is given, then, for each item X, every neighbor that has a similarity value greater than or equal to the cutoff will be returned in the neighbor list for X.

Value

The return value is a list with the following components:

- `indexes` - index values of nearest neighbors, for each item. If `cutoff` is used, this will be a list of lists, otherwise it will be a matrix
- `names` - The names of each item in the set, as returned by cid
- `similarities` - The similarity values of each neighbor to the item for that row. This will also be either a list of lists or a matrix, depending on whether or not `cutoff` was used. Each similarity values corresponds to the id number in the same position in the indexes entry

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See `clusterJP` for Jarvis-Patrick Clustering

Examples

```
data(sdfbcl)
apbcl = convSDFtoAP(sdfbcl)
nnm = NNeighbors(apbcl, cutoff = 0.5)
```

OptAA3d — *OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)*

Description

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

Details

OptAA3d.sdf - 20 Amino Acids Optimized with MOE 2011.10 (Semiempirical AM1)

Examples

```
# This operation requires the rcdk package
# require(rcdk)
# optaa3d = load.molecules(system.file('sysdata/OptAA3d.sdf', package = 'Rcpi'))
# view.molecule.2d(optaa3d[[1]])  # view the first AA
```

`parGOSim` *Protein/DNA Sequence Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein/DNA Sequence Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
parGOSim(golist, type = c("go", "gene"), ont = "MF", organism = "human",
  measure = "Resnik", combine = "BMA")
```

Arguments

`golist`	A character vector, each component contains a character vector of GO terms or one Entrez Gene ID.
`type`	Input type of `golist`, `'go'` for GO Terms, `'gene'` for gene ID.
`ont`	Default is `'MF'`, could be one of `'MF'`, `'BP'`, or `'CC'` subontologies.
`organism`	Default is `'human'`, could be one of `'anopheles'`, `'arabidopsis'`, `'bovine'`, `'canine'`, `'chicken'`, `'chimp'`, `'coelicolor'`, `'ecolik12'`, `'ecsakai'`, `'fly'`, `'human'`, `'malaria'`, `'mouse'`, `'pig'`, `'rat'`, `'rhesus'`, `'worm'`, `'xenopus'`, `'yeast'` or `'zebrafish'`.
`measure`	Default is `'Resnik'`, could be one of `'Resnik'`, `'Lin'`, `'Rel'`, `'Jiang'` or `'Wang'`.
`combine`	Default is `'BMA'`, could be one of `'max'`, `'average'`, `'rcmax'` or `'BMA'` for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates protein/DNA sequence similarity based on Gene Ontology (GO) similarity.

Value

A n x n similarity matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

See Also

See `twoGOSim` for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs. See `parSeqSim` for paralleled protein/DNA similarity calculation based on Smith-Waterman local alignment.

Examples

```
# Be careful when testing this since it involves GO similarity computation
# and might produce unpredictable results in some environments

require(GOSemSim)
require(org.Hs.eg.db)

# by GO Terms
go1 = c('GO:0005215', 'GO:0005488', 'GO:0005515', 'GO:0005625', 'GO:0005802', 'GO:0005905')  # AP4B1
go2 = c('GO:0005515', 'GO:0005634', 'GO:0005681', 'GO:0008380', 'GO:0031202')  # BCAS2
go3 = c('GO:0003735', 'GO:0005622', 'GO:0005840', 'GO:0006412')  # PDE4DIP
glist = list(go1, go2, go3)
gsimmat1 = parGOSim(glist, type = 'go', ont = 'CC')
print(gsimmat1)

# by Entrez gene id
genelist = list(c('150', '151', '152', '1814', '1815', '1816'))
gsimmat2 = parGOSim(genelist, type = 'gene')
print(gsimmat2)
```

`parSeqSim` *Parallellized Protein/DVA Sequence Similarity Calculation based on Sequence Alignment*

Description

Parallellized Protein/DNA Sequence Similarity Calculation based on Sequence Alignment

Usage

```
parSeqSim(protlist, cores = 2, type = "local", submat = "BLOSUM62")
```

Arguments

`protlist`	A length `n` list containing `n` protein sequences, each component of the list is a character string, storing one protein sequence. Unknown sequences should be represented as `''`.
`cores`	Integer. The number of CPU cores to use for parallel execution, default is `2`. Users could use the `detectCores()` function in the `parallel` package to see how many cores they could use.
`type`	Type of alignment, default is `'local'`, could be `'global'` or `'local'`, where `'global'` represents Needleman-Wunsch global alignment; `'local'` represents Smith-Waterman local alignment.
`submat`	Substitution matrix, default is `'BLOSUM62'`, could be one of `'BLOSUM45'`, `'BLOSUM50'`, `'BLOSUM62'`, `'BLOSUM80'`, `'BLOSUM100'`, `'PAM30'`, `'PAM40'`, `'PAM70'`, `'PAM120'`, `'PAM250'`.

Details

This function implemented the parallellized version for calculating protein/DNA sequence similarity based on sequence alignment.

Value

A n x n similarity matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

See Also

See twoSeqSim for protein sequence alignment for two protein/DNA sequences. See parGOSim for protein/DNA similarity calculation based on Gene Ontology (GO) semantic similarity.

Examples

```
# Be careful when testing this since it involves parallelisation
# and might produce unpredictable results in some environments

require(Biostrings)
require(foreach)
require(doParallel)

s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
s2 = readFASTA(system.file('protseq/P08218.fasta', package = 'BioMedR'))[[1]]
s3 = readFASTA(system.file('protseq/P10323.fasta', package = 'BioMedR'))[[1]]
s4 = readFASTA(system.file('protseq/P20160.fasta', package = 'BioMedR'))[[1]]
s5 = readFASTA(system.file('protseq/Q9NZP8.fasta', package = 'BioMedR'))[[1]]
plist = list(s1, s2, s3, s4, s5)
psimmat = parSeqSim(plist, cores = 2, type = 'local', submat = 'BLOSUM62')
print(psimmat)
s11 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[1]]
s21 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[2]]
s31 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[3]]
s41 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[4]]
s51 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[5]]
plist1 = list(s11, s21, s31, s41, s51)
psimmat1 = parSeqSim(plist1, cores = 2, type = 'local', submat = 'BLOSUM62')
print(psimmat1)
```

---

plotStructure *Plots compound structure(s) for molecules stored in SDF and SDFset containers*

Description

Plots compound structure(s) for molecules stored in SDF and SDFset containers.

Usage

```
plotStructure(sdf, atomcex = 1.2, atomnum = FALSE,
  no_print_atoms = c("C"), noHbonds = TRUE, bondspacer = 0.12,
  colbonds = NULL, bondcol = "red", ...)
```

Arguments

sdf	Object of class SDF
atomcex	Font size for atom labels
atomnum	If TRUE, then the atom numbers are included in the plot. They are the position numbers of each atom in the atom block of an SDF.
no_print_atoms	Excludes specified atoms from being plotted.
noHbonds	If TRUE, then the C-hydrogens and their bonds - explicitly defined in an SDF - are excluded from the plot.
bondspacer	Numeric value specifying the plotting distance for double/triple bonds.
colbonds	Highlighting of subgraphs in main structure by providing a numeric vector of atom numbers, here position index in atom block. The bonds of connected atoms will be plotted in the color provided under bondcol.
bondcol	A character or numeric vector of length one to specify the color to use for substructure highlighting under colbonds.
...	Arguments to be passed to/from other methods.

Details

The function plotStructure depicts a single 2D compound structure based on the XY-coordinates specified in the atom block of an SDF. The functions depend on the availability of the XY-coordinates in the source SD file and only 2D (not 3D) representations are plotted correctly.

Value

Prints summary of SDF/SDFset to screen and plots their structures to graphics device.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

...

Examples

```
data(sdfbcl)

plotStructure(sdfbcl[[1]])

plotStructure(sdf = sdfbcl[[2]], atomcex = 1.2, atomnum = FALSE,
             no_print_atoms = c("C"), noHbonds = TRUE, bondspacer = 0.08)
```

pls.cv *The Cross-Validation of Classification and Regression models using Partial Least Squares*

Description

The Cross-Validation of Classification and Regression models using Partial Least Squares

Usage

```
pls.cv(xtr, ytr, cv.fold = 5, maxcomp = NULL)
```

Arguments

xtr	A data frame or a matrix of predictors.
ytr	A response vector. If a factor, classification is assumed, otherwise regression is assumed.
cv.fold	The fold, the defalut is 5.
maxcomp	Maximum number of components included within the models, if not specified, default is the variable (column) numbers in x.

Details

This function performs k-fold cross validation for partial least squares regression and classification.

Value

the retrun a list containing four components:

- plspred - the predicted values of the input data based on cross-validation
- Error - error for all samples
- RMSECV - Root Mean Square Error for cross-validation
- Q2 - R2 for cross-validation

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

See Also

See rf.cv for the Cross-Validation of Classification and Regression models using Random Forest

Examples

```
training = read.csv(system.file('sysdata/training2.csv', package = 'BioMedR'), header = TRUE)
y = training[, 1]
x = training[, -1]
pls.tr <- pls.cv(x, y)
```

property *Calculates Atom Additive logP and Molar Refractivity Values Descriptor*

Description

Calculates Atom Additive logP and Molar Refractivity Values Descriptor

Calculates the Sum of the Atomic Polarizabilities Descriptor

Calculates the Descriptor that Describes the Sum of the Absolute Value of the Difference between Atomic Polarizabilities of All Bonded Atoms in the Molecule

Descriptor that Calculates the Number of Hydrogen Bond Acceptors

Descriptor that Calculates the Number of Hydrogen Bond Donors

Descriptor that Calculates the Number Failures of the Lipinski's Rule Of Five

Descriptor of Topological Polar Surface Area Based on Fragment Contributions (TPSA)

Descriptor that Calculates the Total Weight of Atoms

Descriptor that Calculates the Prediction of logP Based on the Atom-Type Method Called XLogP

Usage

```
extrDrugALOGP(molecules, silent = TRUE)

extrDrugApol(molecules, silent = TRUE)

extrDrugBPol(molecules, silent = TRUE)

extrDrugHBondAcceptorCount(molecules, silent = TRUE)

extrDrugHBondDonorCount(molecules, silent = TRUE)

extrDrugRuleOfFive(molecules, silent = TRUE)

extrDrugTPSA(molecules, silent = TRUE)

extrDrugWeight(molecules, silent = TRUE)

extrDrugLogP(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates ALOGP (Ghose-Crippen LogKow) and the Ghose-Crippen molar refractivity as described by Ghose, A.K. and Crippen, G.M. Note the underlying code in CDK assumes that aromaticity has been detected before evaluating this descriptor. The code also expects that the molecule

will have hydrogens explicitly set. For SD files, this is usually not a problem since hydrogens are explicit. But for the case of molecules obtained from SMILES, hydrogens must be made explicit.

Calculates the sum of the atomic polarizabilities (including implicit hydrogens) descriptor. Polarizabilities are taken from http://www.sunysccc.edu/academic/mst/ptable/p-table2.htm.

This descriptor calculates the sum of the absolute value of the difference between atomic polarizabilities of all bonded atoms in the molecule (including implicit hydrogens) with polarizabilities taken from http://www.sunysccc.edu/academic/mst/ptable/p-table2.htm. This descriptor assumes 2-centered bonds.

This descriptor calculates the number of hydrogen bond acceptors using a slightly simplified version of the PHACIR atom types. The following groups are counted as hydrogen bond acceptors: any oxygen where the formal charge of the oxygen is non-positive (i.e. formal charge <= 0) except

1. an aromatic ether oxygen (i.e. an ether oxygen that is adjacent to at least one aromatic carbon)
2. an oxygen that is adjacent to a nitrogen

and any nitrogen where the formal charge of the nitrogen is non-positive (i.e. formal charge <= 0) except a nitrogen that is adjacent to an oxygen.

This descriptor calculates the number of hydrogen bond donors using a slightly simplified version of the PHACIR atom types (http://www.chemie.uni-erlangen.de/model2001/abstracts/rester.html). The following groups are counted as hydrogen bond donors:

- Any-OH where the formal charge of the oxygen is non-negative (i.e. formal charge >= 0)
- Any-NH where the formal charge of the nitrogen is non-negative (i.e. formal charge >= 0)

This descriptor calculates the number failures of the Lipinski's Rule Of Five: http://en.wikipedia.org/wiki/Lipinski%27s_Rule_of_Five.

Calculate the descriptor of topological polar surface area based on fragment contributions (TPSA).

This descriptor calculates the molecular weight.

Prediction of logP based on the atom-type method called XLogP.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns three columns named `ALogP`, `ALogp2` and `AMR`.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Ghose, A.K. and Crippen, G.M. , Atomic physicochemical parameters for three-dimensional structure-directed quantitative structure-activity relationships. I. Partition coefficients as a measure of hydrophobicity, Journal of Computational Chemistry, 1986, 7:565-577.

Ghose, A.K. and Crippen, G.M. , Atomic physicochemical parameters for three-dimensional-structure-directed quantitative structure-activity relationships. 2. Modeling dispersive and hydrophobic interactions, Journal of Chemical Information and Computer Science, 1987, 27:21-35.

Ertl, P., Rohde, B., & Selzer, P. (2000). Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. Journal of medicinal chemistry, 43(20), 3714-3717.

Wang, R., Fu, Y., and Lai, L., A New Atom-Additive Method for Calculating Partition Coefficients, Journal of Chemical Information and Computer Sciences, 1997, 37:615-621.

Wang, R., Gao, Y., and Lai, L., Calculating partition coefficient by atom-additive method, Perspectives in Drug Discovery and Design, 2000, 19:47-66.

Examples

```
# Calculates Atom Additive logP and Molar Refractivity Values Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugALOGP(mol)
head(dat)
# Calculates the Sum of the Atomic Polarizabilities Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugApol(mol)
head(dat)
# the Sum of the Absolute Value of the Difference between Atomic
# Polarizabilities of All Bonded Atoms in the Molecule
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugBPol(mol)
head(dat)
# Calculates the Number of Hydrogen Bond Acceptors
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugHBondAcceptorCount(mol)
head(dat)
# Calculates the Number of Hydrogen Bond Donors
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugHBondDonorCount(mol)
head(dat)
# Calculates the Number Failures of the Lipinski's Rule Of Five
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugRuleOfFive(mol)
head(dat)
# Descriptor of Topological Polar Surface Area Based on
# Fragment Contributions (TPSA)
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugTPSA(mol)
head(dat)
# Calculates the Total Weight of Atoms
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugWeight(mol)
head(dat)
# Calculates the Prediction of logP
# Based on the Atom-Type Method Called XLogP
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugLogP(mol)
head(dat)
```

readFASTA *Read Protein/DNA Sequences in FASTA Format*

Description

Read Protein/DNA Sequences in FASTA Format

Usage

```
readFASTA(file = system.file("protseq/P00750.fasta", package = "BioMedR"),
  legacy.mode = TRUE, seqonly = FALSE)
```

Arguments

file	The name of the file which the sequences in fasta format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, getwd. The default here is to read the P00750.fasta file which is present in the protseq directory of the BioMedR package.
legacy.mode	If set to TRUE, lines starting with a semicolon ';' are ignored. Default value is TRUE.
seqonly	If set to TRUE, only sequences as returned without attempt to modify them or to get their names and annotations (execution time is divided approximately by a factor 3). Default value is FALSE.

Details

This function reads protein sequences in FASTA format.

Value

The result character vector

Note

Note that any different sets of instances (chunklets), e.g. 1, 3, 7 and 4, 6, might belong to the same class and might belong to different classes.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, **85**: 2444-2448

See Also

See readPDB for reading protein sequences in PDB format.

Examples

```
P00750 = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))
```

readMolFromSDF *Read Molecules from SDF Files and Return Parsed Java Molecular Object*

Description

Read Molecules from SDF Files and Return Parsed Java Molecular Object

Usage

```
readMolFromSDF(sdffile)
```

Arguments

sdffile	Character vector, containing SDF file location(s).

Details

This function reads molecules from SDF files and return parsed Java molecular object needed by `extrDrug...` functions.

Value

A list, containing parsed Java molecular object.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `readMolFromSmi` for reading molecules by SMILES string and returning parsed Java molecular object.

Examples

```
mol  = readMolFromSDF(system.file('compseq/DB00859.sdf', package = 'BioMedR'))
mols = readMolFromSDF(c(system.file('compseq/DB00859.sdf', package = 'BioMedR'),
                        system.file('compseq/DB00860.sdf', package = 'BioMedR')))
```

`readMolFromSmi` *Read Molecules from SMILES Files and Return Parsed Java Molecular Object or Plain Text List*

Description

Read Molecules from SMILES Files and Return Parsed Java Molecular Object or Plain Text List

Usage

```
readMolFromSmi(smifile, type = c("mol", "text"))
```

Arguments

`smifile`	Character vector, containing SMILES file location(s).
`type`	`'mol'` or `'text'`. `'mol'` returns parsed Java molecular object, used for `'text'` returns (plain-text) character string list. For common molecular descriptors and fingerprints, use `'mol'`. For descriptors and fingerprints calculated by OpenBabel, i.e. functions named `extrDrugOB...()` , use `'text'`.

Details

This function reads molecules from SMILES strings and return parsed Java molecular object or plain text list needed by `extrDrug...()` functions.

Value

A list, containing parsed Java molecular object or character strings.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

See Also

See `readMolFromSDF` for reading molecules from SDF files and returning parsed Java molecular object.

Examples

```
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol1 = readMolFromSmi(smi, type = 'mol')
mol2 = readMolFromSmi(smi, type = 'text')
```

readPDB *Read Protein Sequences in PDB Format*

Description

Read Protein Sequences in PDB Format

Usage

```
readPDB(file = system.file("protseq/4HHB.pdb", package = "BioMedR"))
```

Arguments

file	The name of the file which the sequences in PDB format are to be read from. If it does not contain an absolute or relative path, the file name is relative to the current working directory, getwd. The default here is to read the 4HHB.PDB file which is present in the protseq directory of the BioMedR package.

Details

This function reads protein sequences in PDB (Protein Data Bank) format, and return the amino acid sequences represented by single-letter code.

Value

A character vector, representing the amino acid sequence of the single-letter code.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description, Version 3.30. Accessed 2013-06-26. ftp://ftp.wwpdb.org/pub/pdb/doc/format_descriptions/Format_v33_Letter.pdf

See Also

See readFASTA for reading protein sequences in FASTA format.

Examples

```
Seq1atp = readPDB(system.file('protseq/1atp.pdb', package = 'BioMedR'))
Seq1atp
```

revchars *The Reverse chars*

Description

The Reverse chars

Usage

```
revchars(x)
```

Arguments

x	the input data, which should be a string.

Details

This function calculates Reverse chars

Value

A vector

Note

if the user defined physicochemical indices have not been normalized, it should be normalized.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

Examples

```
x = 'GACTGAACTGCACTTTGGTTTCATATTATTTGCTC'
revchars(x)
```

rf.cv *The Cross-Validation of Classification and Regression models using Random Forest*

Description

The Cross-Validation of Classification and Regression models using Random Forest

Usage

```
rf.cv(xtr, ytr, cv.fold = 5, type = "regression", trees = 500,
  mtrysize = 10)
```

Arguments

xtr	A data frame or a matrix of predictors.
ytr	A response vector. If a factor, classification is assumed, otherwise regression is assumed.
cv.fold	The fold, the defalut is 5.
type	method type.
trees	Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
mtrysize	Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in xtr) and regression (p/3)

Details

rf.cv implements Breiman's random forest algorithm for classification and regression. here we use it to make a k-fold cross-validation

Value

if type is regression, the retrun a list containing four components:

- RFpred - the predicted values of the input data based on cross-validation
- Error - error for all samples
- RMSECV - Root Mean Square Error for cross-validation
- Q2 - R2 for cross-validation

if type is classification, the retrun a list containing four components:

- table - confusion matrix
- ACC - accuracy
- SE - sensitivity
- SP - specifivity
- F1 - a measure of a test's accuracy.
- MCC - Mathews correlation coefficient
- RFPred - the predicted values
- prob - the predicted probability values

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Breiman, L. (2001), *Random Forests*, Machine Learning 45(1), 5-32.

See Also

See pls.cv for the Cross-Validation of Classification and Regression models using PLS

Examples

```
training = read.csv(system.file('sysdata/training2.csv', package = 'BioMedR'), header = TRUE)
y = training[, 1]
x = training[, -1]
rf.tr <- rf.cv(x, y)
```

`rf.fs` *Random Forest Cross-Valdidation for feature selection*

Description

Random Forest Cross-Valdidation for feature selection

Usage

```
rf.fs(trainx, trainy, cv.fold = 5, scale = "log", step = 0.5,
  mtry = function(p) max(1, floor(sqrt(p))), recursive = FALSE)
```

Arguments

`trainx`	matrix or data frame containing columns of predictor variables
`trainy`	vector of response, must have length equal to the number of rows in `trainx`
`cv.fold`	The fold, the defalut is 5.
`scale`	If `"log"`, reduce a fixed proportion (`step`) of variables at each step, otherwise reduce `step` variables at a time
`step`	If `log=TRUE`, the fraction of variables to remove at each step, else remove this many variables at a time
`mtry`	A function of number of remaining predictor variables to use as the `mtry` parameter in the `randomForest` call
`recursive`	Whether variable importance is (re-)assessed at each step of variable reduction

Details

This function shows the cross-validated prediction performance of models with sequentially reduced number of predictors (ranked by variable importance) via a nested cross-validation procedure.

Value

A list with the following three components::

- `n.var` - vector of number of variables used at each step
- `error.cv` - corresponding vector of error rates or MSEs at each step
- `res` - list of n.var components, each containing the feature importance values from the cross-validation

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>

References

Svetnik, V., Liaw, A., Tong, C. and Wang, T., Application of Breiman's Random Forest to Modeling Structure-Activity Relationships of Pharmaceutical Molecules, MCS 2004, Roli, F. and Windeatt, T. (Eds.) pp. 334-343.

See Also

See `rf.cv` for the Cross-Validation of Classification and Regression models using Random Forest

Examples

```
training = read.csv(system.file('sysdata/training1.csv', package = 'BioMedR'), header = TRUE)
y = training[, 1]
x = training[, -1]
result = rf.fs(x, y)
```

---

sdfbcl *SD file in* `SDFset` *object*

Description

SD file in `SDFset` object

Usage

```
data(sdfbcl)
```

Details

`SDFset` object for the 50 SDF.

Examples

```
data(sdfbcl)
```

---

searchDrug *Parallelized Drug Molecule Similarity Search by Molecular Fingerprints Similarity or Maximum Common Substructure Search*

Description

Parallelized Drug Molecule Similarity Search by Molecular Fingerprints Similarity or Maximum Common Substructure Search

Usage

```
searchDrug(mol, moldb, cores = 2, method = c("fp", "mcs"),
  fptype = c("standard", "extended", "graph", "hybrid", "maccs", "estate",
  "pubchem", "kr", "shortestpath", "fp2", "fp3", "fp4"), fpsim = c("tanimoto",
  "euclidean", "cosine", "dice", "hamming"), mcssim = c("tanimoto",
  "overlap"), ...)
```

Arguments

mol	The query molecule. The location of a sdf file containing one molecule.
moldb	The molecule database. The location of a sdf file containing all the molecules to be searched with.
cores	Integer. The number of CPU cores to use for parallel search, default is 2. Users could use the detectCores() function in the parallel package to see how many cores they could use.
method	'fp' or 'mcs'. Search by molecular fingerprints or by maximum common substructure searching.
fptype	The fingerprint type, only available when method = 'fp'. BioMedR supports 13 types of fingerprints, including 'standard', 'extended', 'graph', 'hybrid', 'maccs', 'estate', 'pubchem', 'kr', 'shortestpath', 'fp2', 'fp3', 'fp4'.
fpsim	Similarity measure type for fingerprint, only available when method = 'fp'. Including 'tanimoto', 'euclidean', 'cosine', 'dice' and 'hamming'. See calcDrugFPSim for details.
mcssim	Similarity measure type for maximum common substructure search, only available when method = 'mcs'. Including 'tanimoto' and 'overlap'.
...	Other possible parameter for maximum common substructure search, see calcDrugMCSSim for available options.

Details

This function does compound similarity search derived by various molecular fingerprints with various similarity measures or derived by maximum common substructure search. This function runs for a query compound against a set of molecules.

Value

Named numerical vector. With the decreasing similarity value of the molecules in the database.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
mol = system.file('compseq/example.sdf', package = 'BioMedR')
# DrugBank ID DB00530: Erlotinib
moldb = system.file('compseq/bcl.sdf', package = 'BioMedR')
# Database composed by searching 'tyrphostin' in PubChem and filtered by Lipinski's Rule of Five
searchDrug(mol, moldb, cores = 4, method = 'fp', fptype = 'maccs', fpsim = 'hamming')
searchDrug(mol, moldb, cores = 4, method = 'fp', fptype = 'fp2', fpsim = 'tanimoto')
searchDrug(mol, moldb, cores = 4, method = 'mcs', mcssim = 'tanimoto')
```

segProt *Protein Sequence Segmentation*

Description

Protein Sequence Segmentation

Usage

```
segProt(x, aa = c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I", "L", "K",
  "M", "F", "P", "S", "T", "W", "Y", "V"), k = 7)
```

Arguments

x	A character vector, as the input protein sequence.
aa	A character, the amino acid type. one of 'A', 'R', 'N', 'D', 'C', 'E', 'Q', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V'.
k	A positive integer, specifys the window size (half of the window), default is 7.

Details

This function extracts the segmentations from the protein sequence.

Value

A named list, each component contains one of the segmentations (a character string), names of the list components are the positions of the specified amino acid in the sequence.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

Examples

```
x = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
segProt(x, aa = 'R', k = 5)
```

topology *Topological Descriptor Characterizing the Carbon Connectivity in Terms of Hybridization*

Description

Topological Descriptor Characterizing the Carbon Connectivity in Terms of Hybridization

Calculates the Eccentric Connectivity Index Descriptor

Calculates the FMF Descriptor

Calculate Complexity of a System

Calculate Molecular Distance Edge (MDE) Descriptors for C, N and O

Descriptor that Calculates the Petitjean Number of a Molecule

Descriptor that Calculates the Petitjean Shape Indices

Descriptor that Calculates the Volume of A Molecule

Descriptor that Calculates the Vertex Adjacency Information of A Molecule

Descriptor that Calculates the Weighted Path (Molecular ID)

Descriptor that Calculates Wiener Path Number and Wiener Polarity Number

Descriptor that Calculates the Sum of the Squared Atom Degrees of All Heavy Atoms

Usage

```
extrDrugCarbonTypes(molecules, silent = TRUE)

extrDrugECI(molecules, silent = TRUE)

extrDrugFMF(molecules, silent = TRUE)

extrDrugFragmentComplexity(molecules, silent = TRUE)

extrDrugMDE(molecules, silent = TRUE)

extrDrugPetitjeanNumber(molecules, silent = TRUE)

extrDrugPetitjeanShapeIndex(molecules, silent = TRUE)

extrDrugVABC(molecules, silent = TRUE)

extrDrugVAdjMa(molecules, silent = TRUE)

extrDrugWeightedPath(molecules, silent = TRUE)

extrDrugWienerNumbers(molecules, silent = TRUE)

extrDrugZagrebIndex(molecules, silent = TRUE)
```

Arguments

molecules	Parsed molucule object.
silent	Logical. Whether the calculating process should be shown or not, default is TRUE.

Details

Calculates the carbon connectivity in terms of hybridization. The function calculates 9 descriptors in the following order:

- `C1SP1` - triply hound carbon bound to one other carbon
- `C2SP1` - triply bound carbon bound to two other carbons
- `C1SP2` - doubly hound carbon bound to one other carbon
- `C2SP2` - doubly bound carbon bound to two other carbons
- `C3SP2` - doubly bound carbon bound to three other carbons
- `C1SP3` - singly bound carbon bound to one other carbon
- `C2SP3` - singly bound carbon bound to two other carbons
- `C3SP3` - singly bound carbon bound to three other carbons
- `C4SP3` - singly bound carbon bound to four other carbons

Eccentric Connectivity Index (ECI) is a topological descriptor combining distance and adjacency information. This descriptor is described by Sharma et al. and has been shown to correlate well with a number of physical properties. The descriptor is also reported to have good discriminatory ability. The eccentric connectivity index for a hydrogen supressed molecular graph is given by

$$x_i^c = \sum_{i=1}^{n} E(i)V(i)$$

where E(i) is the eccentricity of the i-th atom (path length from the i-th atom to the atom farthest from it) and V(i) is the vertex degree of the i-th atom.

Calculates the FMF descriptor characterizing molecular complexity in terms of its Murcko framework. This descriptor is the ratio of heavy atoms in the framework to the total number of heavy atoms in the molecule. By definition, acyclic molecules which have no frameworks, will have a value of 0. Note that the authors consider an isolated ring system to be a framework (even though there is no linker).

This descriptor calculates the complexity of a system. The complexity is defined in Nilakantan, R. et al. as:

$$C = abs(B^2 - A^2 + A) + \frac{H}{100}$$

where C is complexity, A is the number of non-hydrogen atoms, B is the number of bonds and H is the number of heteroatoms.

This descriptor calculates the 10 molecular distance edge (MDE) descriptor described in Liu, S., Cao, C., & Li, Z, and in addition it calculates variants where O and N are considered.

This descriptor calculates the Petitjean number of a molecule. According to the Petitjean definition, the eccentricity of a vertex corresponds to the distance from that vertex to the most remote vertex in the graph.

The distance is obtained from the distance matrix as the count of edges between the two vertices. If `r(i)` is the largest matrix entry in row `i` of the distance matrix `D`, then the radius is defined as the smallest of the `r(i)`. The graph diameter `D` is defined as the largest vertex eccentricity in the graph. (`http://www.edusoft-lc.com/molconn/manuals/400/chaptwo.html`)

The topological and geometric shape indices described Petitjean and Bath et al. respectively. Both measure the anisotropy in a molecule.

This descriptor calculates the volume of a molecule.

Vertex adjacency information (magnitude): $1+\log_2^m$ where m is the number of heavy-heavy bonds. If m is zero, then `0` is returned.

This descriptor calculates the weighted path (molecular ID) described by Randic, characterizing molecular branching. Five descriptors are calculated, based on the implementation in the ADAPT software package. Note that the descriptor is based on identifying all paths between pairs of atoms and so is NP-hard. This means that it can take some time for large, complex molecules.

This descriptor calculates the Wiener numbers, including the Wiener Path number and the Wiener Polarity Number. Wiener path number: half the sum of all the distance matrix entries; Wiener polarity number: half the sum of all the distance matrix entries with a value of 3.

Zagreb index: the sum of the squares of atom degree over all heavy atoms `i`.

Value

A data frame, each row represents one of the molecules, each column represents one feature. This function returns 9 columns named `C1SP1`, `C2SP1`, `C1SP2`, `C2SP2`, `C3SP2`, `C1SP3`, `C2SP3`, `C3SP3` and `C4SP3`.

WTPT `WTPT.1`, `WTPT.2`, `WTPT.3`, `WTPT.4`, `WTPT.5`:

- `WTPT.1` - molecular ID
- `WTPT.2` - molecular ID / number of atoms
- `WTPT.3` - sum of path lengths starting from heteroatoms
- `WTPT.4` - sum of path lengths starting from oxygens
- `WTPT.5` - sum of path lengths starting from nitrogens

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://r2s.name>

References

Sharma, V. and Goswami, R. and Madan, A.K. (1997), Eccentric Connectivity Index: A Novel Highly Discriminating Topological Descriptor for Structure-Property and Structure-Activity Studies, Journal of Chemical Information and Computer Sciences, 37:273-282

Yang, Y., Chen, H., Nilsson, I., Muresan, S., & Engkvist, O. (2010). Investigation of the relationship between topology and selectivity for druglike molecules. Journal of medicinal chemistry, 53(21), 7709-7714.

Nilakantan, R. and Nunn, D.S. and Greenblatt, L. and Walker, G. and Haraki, K. and Mobilio, D., A family of ring system-based structural fragments for use in structure-activity studies: database mining and recursive partitioning., Journal of chemical information and modeling, 2006, 46:1069-1077

Liu, S., Cao, C., & Li, Z. (1998). Approach to estimation and prediction for normal boiling point (NBP) of alkanes based on a novel molecular distance-edge (MDE) vector, lambda. Journal of chemical information and computer sciences, 38(3), 387-394.

Petitjean, M., Applications of the radius-diameter diagram to the classification of topological and geometrical shapes of chemical compounds, Journal of Chemical Information and Computer Science, 1992, 32:331-337

Bath, P.A. and Poirette, A.R. and Willet, P. and Allen, F.H. , The Extent of the Relationship between the Graph-Theoretical and the Geometrical Shape Coefficients of Chemical Compounds, Journal of Chemical Information and Computer Science, 1995, 35:714-716.

Randic, M., On molecular identification numbers (1984). Journal of Chemical Information and Computer Science, 24:164-175.

Wiener, H. (1947). Structural determination of paraffin boiling points. Journal of the American Chemical Society, 69(1), 17-20.

Examples

```
# Topological Descriptor Characterizing the Carbon Connectivity
# in Terms of Hybridization
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugCarbonTypes(mol)
head(dat)
# Calculates the Eccentric Connectivity Index Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugECI(mol)
head(dat)
# Calculates the FMF Descriptor
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugFMF(mol)
head(dat)
# Calculate Complexity of a System
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugFragmentComplexity(mol)
head(dat)
# Calculate Molecular Distance Edge (MDE) Descriptors for C, N and O
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugMDE(mol)
head(dat)
# Calculates the Petitjean Number of a Molecule
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugPetitjeanNumber(mol)
head(dat)
# Calculates the Petitjean Shape Indices
sdf = system.file('sysdata/test.sdf', package = 'BioMedR')
mol = readMolFromSDF(sdf)
dat = extrDrugPetitjeanShapeIndex(mol)
head(dat)
# Calculates the Volume of A Molecule
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugVABC(mol)
head(dat)
# Calculates the Vertex Adjacency Information of A Molecule
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugVAdjMa(mol)
head(dat)
# Calculates the Weighted Path (Molecular ID)
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugWeightedPath(mol)
```

```
head(dat)
# Calculates Wiener Path Number and Wiener Polarity Number
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugWienerNumbers(mol)
head(dat)
# Calculates the Sum of the Squared Atom Degrees
# of All Heavy Atoms
smi = system.file('vignettedata/test.smi', package = 'BioMedR')
mol = readMolFromSmi(smi, type = 'mol')
dat = extrDrugZagrebIndex(mol)
head(dat)
```

`twoGOSim` *Protein/DNA Similarity Calculation based on Gene Ontology (GO) Similarity*

Description

Protein/DNA Similarity Calculation based on Gene Ontology (GO) Similarity

Usage

```
twoGOSim(id1, id2, type = c("go", "gene"), ont = "MF", organism = "human",
  measure = "Resnik", combine = "BMA")
```

Arguments

id1	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
id2	A character vector. length > 1: each element is a GO term; length = 1: the Entrez Gene ID.
type	Input type of id1 and id2, 'go' for GO Terms, 'gene' for gene ID.
ont	Default is 'MF', could be one of 'MF', 'BP', or 'CC' subontologies.
organism	Default is 'human', could be one of 'anopheles', 'arabidopsis', 'bovine', 'canine', 'chicken', 'chimp', 'coelicolor', 'ecolik12', 'ecsakai', 'fly', 'human', 'malaria', 'mouse', 'pig', 'rat', 'rhesus', 'worm', 'xenopus', 'yeast' or 'zebrafish'.
measure	Default is 'Resnik', could be one of 'Resnik', 'Lin', 'Rel', 'Jiang' or 'Wang'.
combine	Default is 'BMA', could be one of 'max', 'average', 'rcmax' or 'BMA' for combining semantic similarity scores of multiple GO terms associated with protein.

Details

This function calculates the Gene Ontology (GO) similarity between two groups of GO terms or two Entrez gene IDs.

Value

A n x n matrix.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

See Also

See parGOSim for protein similarity calculation based on Gene Ontology (GO) semantic similarity. See parSeqSim for paralleled protein/DNA similarity calculation based on Smith-Waterman local alignment.

Examples

```
# Be careful when testing this since it involves GO similarity computation
# and might produce unpredictable results in some environments

require(GOSemSim)
require(org.Hs.eg.db)

# by GO terms
go1 = c("GO:0004022", "GO:0004024", "GO:0004023")
go2 = c("GO:0009055", "GO:0020037")
gsim1 = twoGOSim(go1, go2, type = 'go', ont = 'MF', measure = 'Wang')
print(gsim1)

# by Entrez gene id
gene1 = '241'
gene2 = '251'
gsim2 = twoGOSim(gene1, gene2, type = 'gene', ont = 'BP', measure = 'Lin')
print(gsim2)
```

---

twoSeqSim *Protein/DNA Sequence Alignment for Two Protein Sequences*

Description

Protein/DNA Sequence Alignment for Two Protein Sequences

Usage

```
twoSeqSim(seq1, seq2, type = "local", submat = "BLOSUM62")
```

Arguments

seq1	A character string, containing one protein sequence.
seq2	A character string, containing another protein sequence.
type	Type of alignment, default is 'local', could be 'global' or 'local', where 'global' represents Needleman-Wunsch global alignment; 'local' represents Smith-Waterman local alignment.

submat — Substitution matrix, default is 'BLOSUM62', could be one of 'BLOSUM45', 'BLOSUM50', 'BLOSUM62', 'BLOSUM80', 'BLOSUM100', 'PAM30', 'PAM40', 'PAM70', 'PAM120', 'PAM250'.

Details

This function implements the sequence alignment between two protein/DNA sequences.

Value

An Biostrings object containing the scores and other alignment information.

Author(s)

Min-feng Zhu <<wind2zhu@163.com>>, Nan Xiao <http://nanx.me>

See Also

See parSeqSim for paralleled pairwise protein similarity calculation based on sequence alignment. See twoGOSim for calculating the GO semantic similarity between two groups of GO terms or two Entrez gene IDs.

Examples

```
# Be careful when testing this since it involves sequence alignment
# and might produce unpredictable results in some environments

require(Biostrings)

s1 = readFASTA(system.file('protseq/P00750.fasta', package = 'BioMedR'))[[1]]
s2 = readFASTA(system.file('protseq/P10323.fasta', package = 'BioMedR'))[[1]]
seqalign = twoSeqSim(s1, s2)
summary(seqalign)
s11 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[1]]
s21 = readFASTA(system.file('dnaseq/hs.fasta', package = 'BioMedR'))[[2]]
seqalign1 = twoSeqSim(s11, s21)
summary(seqalign1)
```

Index